

# Strojové učení 1 (Machine Learning 1)

Matěj Štaif

CTU–FIT

[staifmat@fit.cvut.cz](mailto:staifmat@fit.cvut.cz)

January 1, 2026

## Abstrakt

Tento dokument představuje soubor poznámek pro přípravu na zkoušku z předmětu Strojové učení 1 (BI-ML1) na ČVUT FIT (ZS 2025/26). Text je syntézou informací prezentovaných na přednáškách, autorský jsem jej však zpracoval samostatně. Prohlašuji, že samotný text nebyl generován pomocí LLM. V případě nalezení chyb prosím využijte issues nebo kontakt v záhlaví.

## Obsah

|  |           |
|--|-----------|
| <b>1 Obecné pojmy</b>  | <b>5</b>  |
| 1.1 Klasifikace a regrese (Classification and Regression)          | 5         |
| 1.2 Supervizované a nesupervizované učení                          | 6         |
| 1.3 Nominální a ordinální příznaky                                 | 6         |
| 1.4 Singulární a regulární matice                                  | 6         |
| 1.5 Parametry vs. hyperparametry                                   | 7         |
| 1.6 Parametrické vs. neparametrické modely                         | 7         |
| 1.7 Přeúčení vs. podučení (Overfitting vs. Underfitting)           | 8         |
| <b>2 Rozhodovací stromy (Decision Trees)</b>                       | <b>8</b>  |
| 2.1 Rozhodovací stromy: výhody a nevýhody                          | 8         |
| 2.2 Rozhodovací stromy: konstrukce                                 | 9         |
| 2.3 ID3 (Iterative Dichotomiser 3)                                 | 9         |
| 2.4 Regrese & CART (Classification and Regression Trees)           | 11        |
| 2.5 Hyperparametry rozhodovacích stromů                            | 12        |
| 2.6 Zajímavá videa na YouTube                                      | 13        |
| <b>3 kNN (k-Nearest Neighbor)</b>                                  | <b>13</b> |
| 3.1 kNN: základní popis  | 13        |
| 3.2 kNN: hyperparametry  | 13        |
| 3.3 Normalizace dat  | 14        |
| 3.4 kNN a nominální příznaky                                       | 15        |
| 3.5 Prokletí dimenzionality (The Curse of Dimensionality)          | 15        |
| 3.6 Zajímavá videa na YouTube                                      | 16        |
| <b>4 Lineární regrese (Linear Regression)</b>                      | <b>16</b> |
| 4.1 Metoda nejmenších čtverců (Ordinary Least Squares)             | 16        |
| 4.2 Řešení normální rovnice pomocí SVD rozkladu                    | 18        |
| 4.3 Predikce lineární regrese                                      | 18        |
| 4.4 Závěrečné poznámky   | 19        |
| 4.5 Problém kolinearity  | 19        |
| 4.6 Příklad výpočtu metody nejmenších čtverců                      | 19        |
| 4.6.1 Krok 1: Sestavení matice $\mathbf{X}$ a vektoru $\mathbf{Y}$ | 23        |
| 4.6.2 Krok 2: Výpočet $\mathbf{X}^T \mathbf{X}$                    | 23        |

|           |   |           |
|-----------|---|-----------|
| 4.6.3     | Krok 3: Výpočet $\mathbf{X}^T \mathbf{Y}$   | 23        |
| 4.6.4     | Krok 4: Výpočet inverzní matici $(\mathbf{X}^T \mathbf{X})^{-1}$  | 23        |
| 4.6.5     | Krok 5: Řešení normální rovnice $\hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}$ | 24        |
| 4.6.6     | Krok 6: Výpočet predikcí a reziduí  | 24        |
| 4.6.7     | Shrnutí výsledků  | 24        |
| 4.7       | Geometrická interpretace metody nejmenších čtverců  | 24        |
| 4.8       | Zajímavá videa na YouTube   | 25        |
| <b>5</b>  | <b>Hřebenová regrese (Ridge Regression)</b>   | <b>25</b> |
| 5.1       | Hřebenová regrese: konstrukce   | 26        |
| 5.2       | Modely bázových funkcí  | 27        |
| <b>6</b>  | <b>Statistické vlastnosti modelů</b>  | <b>27</b> |
| 6.1       | Rozklad očekávané chyby modelu  | 27        |
| 6.2       | Bias-variance tradeoff (ilustrace na hřebenové regresi)   | 28        |
| 6.3       | Nestrannost odhadu v lineární regresi metodou nejmenších čtverců  | 29        |
| <b>7</b>  | <b>Logistická regrese</b>   | <b>29</b> |
| 7.1       | Použití pro binární klasifikaci   | 29        |
| 7.2       | Logistická funkce (Sigmoid)   | 29        |
| 7.3       | Způsob výpočtu  | 30        |
| 7.4       | Logistická regrese: vlastnosti  | 30        |
| 7.5       | Maximálně věrohodný odhad (Maximum Likelihood Estimation)   | 30        |
| 7.6       | Závěrečné poznámky  | 32        |
| <b>8</b>  | <b>Ensemble metody (Ensemble Methods)</b>   | <b>33</b> |
| 8.1       | Ensemble metody: základní myšlenka  | 33        |
| 8.2       | Bagging vs. Boosting  | 33        |
| 8.3       | Bagging: náhodné lesy   | 33        |
| 8.4       | Náhodný les: vizualizace  | 34        |
| 8.5       | Bootstrap   | 34        |
| 8.6       | Predikce náhodného lesa   | 34        |
| 8.7       | Hyperparametry náhodných lesů   | 35        |
| 8.8       | Poznámky k náhodným lesům   | 35        |
| 8.9       | Boosting obecně   | 36        |
| 8.10      | AdaBoost (Adaptive Boosting): popis algoritmu   | 36        |
| 8.11      | AdaBoost: predikce  | 37        |
| 8.12      | Poznámky k AdaBoostu  | 37        |
| 8.13      | Zajímavá videa na YouTube   | 37        |
| <b>9</b>  | <b>Evaluace modelů: metriky</b>   | <b>37</b> |
| 9.1       | Úvod k evaluaci modelů  | 37        |
| 9.2       | Ztrátová funkce   | 38        |
| 9.3       | Metriky pro regresi   | 38        |
| 9.4       | Metriky pro klasifikaci   | 38        |
| 9.4.1     | Matice záměn (Confusion Matrix)   | 38        |
| 9.4.2     | Odvozené metriky z matice záměn   | 39        |
| 9.5       | ROC křivka  | 40        |
| 9.6       | AUC (Area Under Curve)  | 40        |
| <b>10</b> | <b>Evaluace modelů: testovací chyba a její odhad</b>  | <b>41</b> |
| 10.1      | Vyhodnocovací scénáře   | 41        |
| 10.2      | Trénovací, validační a testovací množina  | 42        |
| 10.3      | Křížová validace  | 42        |
| 10.4      | Trénovací chyba   | 43        |
| 10.5      | Učení modelu  | 43        |
| 10.6      | Testovací chyba   | 43        |

|   |           |
|---|-----------|
| <b>11 Výběr příznaků (Feature Selection)</b>                    | <b>44</b> |
| 11.1 Úvod do výběru příznaků . . . . .                          | 44        |
| 11.2 Filtrační metody . . . . .                                 | 44        |
| 11.3 Obalové metody . . . . .                                   | 45        |
| 11.4 Vestavěné metody . . . . .                                 | 45        |
| 11.5 Lasso regrese . . . . .                                    | 45        |
| 11.6 Elastic Net regrese . . . . .                              | 47        |
| <b>12 Hierarchické shlukování (Hierarchical Clustering)</b>     | <b>47</b> |
| 12.1 Nesupervizované učení (Unsupervised Learning) . . . . .    | 47        |
| 12.2 Shluková analýza (Cluster Analysis) . . . . .              | 48        |
| 12.2.1 Vzdálenost (metrika) . . . . .                           | 48        |
| 12.2.2 Příklady vzdáleností na $\mathbb{R}^p$ . . . . .         | 49        |
| 12.2.3 Vstupy a výstupy shlukování . . . . .                    | 50        |
| 12.3 Hierarchické shlukování . . . . .                          | 50        |
| 12.3.1 Měření vzdálenosti shluků . . . . .                      | 51        |
| 12.3.2 Vizualizace pomocí dendrogramu . . . . .                 | 51        |
| 12.3.3 Získání shlukování z dendrogramu . . . . .               | 52        |
| 12.3.4 Poznámky k hierarchickému shlukování . . . . .           | 52        |
| <b>13 Shlukování pomocí algoritmu k-means</b>                   | <b>52</b> |
| 13.1 Nesupervizované učení a shluková analýza . . . . .         | 53        |
| 13.2 Shlukování jako optimalizační úloha . . . . .              | 53        |
| 13.3 Souvislost účelové funkce s geometrickým středem . . . . . | 53        |
| 13.4 Algoritmus k-means . . . . .                               | 54        |
| 13.5 Důkaz lokální optimalizace . . . . .                       | 54        |
| 13.6 Ukázka výsledku algoritmu k-means . . . . .                | 55        |
| 13.7 Problematika volby $k$ . . . . .                           | 55        |
| 13.8 Vyhodnocování pomocí Silhouette skóre . . . . .            | 56        |
| <b>14 Shlukování pomocí algoritmu DBSCAN</b>                    | <b>57</b> |
| 14.1 Úvod do shlukování pomocí hustoty . . . . .                | 57        |
| 14.2 DBSCAN: úvodní pojmy . . . . .                             | 58        |
| 14.3 DBSCAN: klíčové body, přímá dosažitelnost . . . . .        | 58        |
| 14.4 DBSCAN: dosažitelnost . . . . .                            | 59        |
| 14.5 DBSCAN: spojenost . . . . .                                | 59        |
| 14.6 DBSCAN: shluky a šum . . . . .                             | 59        |
| 14.7 Poznámky k definici shluků . . . . .                       | 60        |
| 14.8 Abstraktní popis algoritmu DBSCAN . . . . .                | 60        |
| 14.9 DBSCAN: volba parametrů . . . . .                          | 60        |
| 14.10 DBSCAN: ukázka a poznámky . . . . .                       | 61        |
| 14.11 Evaluace pomocí Silhouette skóre . . . . .                | 61        |
| <b>15 Nesupervizované učení, asociační pravidla</b>             | <b>63</b> |
| 15.1 Nesupervizované učení (Unsupervised Learning) . . . . .    | 63        |
| 15.2 Analýza asociačních pravidel . . . . .                     | 64        |
| 15.3 Analýza nákupního košíku . . . . .                         | 64        |
| 15.4 Asociační pravidla: definice . . . . .                     | 64        |
| 15.5 Asociační pravidla: další míry . . . . .                   | 65        |
| 15.6 Asociační pravidla: příklady . . . . .                     | 65        |
| 15.7 Shrnutí asociačních pravidel . . . . .                     | 66        |
| <b>16 Zajímavá videa na YouTube přesahující rámec kurzu</b>     | <b>66</b> |
| 16.1 Vybraná videa z pravděpodobnosti a statistiky . . . . .    | 66        |
| 16.2 Vybraná videa z lineární algebry . . . . .                 | 66        |
| 16.3 Markov Chains . . . . .                                    | 67        |
| 16.4 Hidden Markov Models . . . . .                             | 67        |
| 16.5 Deep Learning . . . . .                                    | 67        |

|  |           |
|--|-----------|
| 16.6 Large Language Models . . . . .   | 67        |
| 16.7 The Black-Scholes-Merton Model . . . . .                                  | 67        |
| 16.8 Principal Component Analysis (PCA) . . . . .                              | 67        |
| 16.9 Support Vector Machines (SVM) . . . . .                                   | 68        |
| <b>17 Souhrn důležitých vzorců</b> . . . . .                                   | <b>68</b> |
| 17.1 Rozhodovací stromy (Decision Trees) . . . . .                             | 68        |
| 17.1.1 Entropie (Entropy) . . . . .  | 68        |
| 17.1.2 Informační zisk (Information Gain) . . . . .                            | 68        |
| 17.1.3 Gini Index (Gini impurity) . . . . .                                    | 68        |
| 17.1.4 Analogie informačního zisku pro regresi . . . . .                       | 68        |
| 17.2 kNN (k-Nearest Neighbor) . . . . .  | 69        |
| 17.2.1 k-normy . . . . .   | 69        |
| 17.2.2 Euklidovská vzdálenost . . . . .  | 69        |
| 17.2.3 Manhattaneská vzdálenost . . . . .                                      | 69        |
| 17.2.4 Min-max normalizace . . . . .   | 69        |
| 17.2.5 Standardizace . . . . .   | 69        |
| 17.3 Lineární regrese (Linear Regression) . . . . .                            | 69        |
| 17.3.1 Residuální součet čtverců (Residual Sum of Squares) . . . . .           | 69        |
| 17.3.2 Gradient $RSS(\mathbf{w})$ . . . . .                                    | 70        |
| 17.3.3 Hessova matice $RSS(\mathbf{w})$ . . . . .                              | 70        |
| 17.3.4 Důkaz pozitivní definitnosti Hessovy matice $RSS(\mathbf{w})$ . . . . . | 70        |
| 17.3.5 Řešení normální rovnice $\hat{\mathbf{w}}$ . . . . .                    | 70        |
| 17.3.6 Řešení normální rovnice pomocí SVD rozkladu . . . . .                   | 71        |
| 17.3.7 Geometrická interpretace metody nejmenších čtverců . . . . .            | 71        |
| 17.4 Hřebenová regrese (Ridge Regression) . . . . .                            | 72        |
| 17.4.1 Regularizovaný reziduální součet čtverců . . . . .                      | 72        |
| 17.4.2 Gradient $RSS_{\lambda}(\mathbf{w})$ . . . . .                          | 72        |
| 17.4.3 Normální rovnice . . . . .  | 72        |
| 17.4.4 Hessova matice $RSS_{\lambda}(\mathbf{w})$ . . . . .                    | 72        |
| 17.4.5 Důkaz pozitivní definitnosti Hessovy matice . . . . .                   | 72        |
| 17.4.6 Řešení normální rovnice $\hat{\mathbf{w}}_{\lambda}$ . . . . .          | 73        |
| 17.4.7 Model bázových funkcí . . . . .   | 73        |
| 17.5 Statistické vlastnosti modelů . . . . .                                   | 73        |
| 17.5.1 Rozklad očekávané chyby . . . . .                                       | 73        |
| 17.5.2 Definice biasu a variance . . . . .                                     | 73        |
| 17.5.3 Bias-variance tradeoff . . . . .  | 73        |
| 17.5.4 Nestrannost OLS . . . . .   | 73        |
| 17.5.5 Důkaz: nestrannost OLS . . . . .  | 74        |
| 17.6 Logistická regrese . . . . .  | 74        |
| 17.6.1 Logistická funkce (sigmoida) . . . . .                                  | 74        |
| 17.6.2 Výpočet $P(Y = 1   \mathbf{x}, \mathbf{w})$ . . . . .                   | 74        |
| 17.6.3 Maximálně věrohodný odhad (Maximum Likelihood Estimation) . . . . .     | 74        |
| 17.7 Ensemble metody (Ensemble Methods) . . . . .                              | 74        |
| 17.7.1 Predikce náhodného lesa (regrese) . . . . .                             | 74        |
| 17.7.2 Predikce náhodného lesa (binární klasifikace) . . . . .                 | 75        |
| 17.7.3 AdaBoost: výpočet $\alpha^{(m)}$ . . . . .                              | 75        |
| 17.7.4 AdaBoost: aktualizace vah . . . . .                                     | 75        |
| 17.8 Evaluace modelů: metriky . . . . .  | 75        |
| 17.8.1 Odhad pravděpodobnosti $\hat{p}$ . . . . .                              | 75        |
| 17.8.2 Ztrátová funkce $L(Y, \hat{p})$ (binary cross-entropy) . . . . .        | 75        |
| 17.8.3 MSE (Mean Squared Error) . . . . .                                      | 75        |
| 17.8.4 RMSE (Root Mean Squared Error) . . . . .                                | 75        |
| 17.8.5 RMSLE (Root Mean Squared Logarithmic Error) . . . . .                   | 76        |
| 17.8.6 MAE (Mean Absolute Error) . . . . .                                     | 76        |
| 17.8.7 $R^2$ (koeficient determinace) . . . . .                                | 76        |
| 17.8.8 Matice záměn (Confusion Matrix) . . . . .                               | 76        |

|   |    |
|---|----|
| 17.8.9 Odvozené metriky z matice záměn  | 76 |
| 17.8.10 ACC (Accuracy)  | 76 |
| 17.8.11 F1 score  | 77 |
| 17.9 Evaluace modelů: testovací chyba a její odhad                                    | 77 |
| 17.9.1 Trénovací chyba  | 77 |
| 17.9.2 Testovací chyba $err_{test}$   | 77 |
| 17.9.3 Cross-validation chyba $\hat{e}$   | 77 |
| 17.10 Výběr příznaků (Feature Selection)  | 77 |
| 17.10.1 Regularizovaný reziduální součet čtverců $RSS_{\lambda}^{Lasso}(\mathbf{w})$  | 77 |
| 17.10.2 Definice řešení Lasso: $\hat{\mathbf{w}}_{\lambda}^{Lasso}$                   | 77 |
| 17.10.3 Elastic Net   | 77 |
| 17.11 Hierarchické shlukování (Hierarchical Clustering)                               | 78 |
| 17.11.1 Euklidovská vzdálenost $L_2$  | 78 |
| 17.11.2 Manhattanská vzdálenost $L_1$   | 78 |
| 17.11.3 Čebyševova vzdálenost $L_{\infty}$  | 78 |
| 17.11.4 Měření vzdálenosti shluků: Metoda nejbližšího souseda (single linkage)        | 78 |
| 17.11.5 Měření vzdálenosti shluků: Metoda nejvzdálenějšího souseda (complete linkage) | 78 |
| 17.11.6 Měření vzdálenosti shluků: Párová vzdálenost (average linkage)                | 78 |
| 17.11.7 Měření vzdálenosti shluků: Wardova metoda                                     | 78 |
| 17.12 Shlukování pomocí algoritmu k-means   | 78 |
| 17.12.1 Účelová funkce pro k-means $G(\mathcal{C})$                                   | 78 |
| 17.12.2 Souvislost účelové funkce s geometrickým středem                              | 79 |
| 17.12.3 Algoritmus k-means  | 79 |
| 17.12.4 Silhouette skóre: vnitřní rozdílnost $a(\mathbf{x})$                          | 79 |
| 17.12.5 Silhouette skóre: sousední rozdílnost $b(\mathbf{x})$                         | 79 |
| 17.12.6 Silhouette skóre bodu $\mathbf{x}$ : $s(\mathbf{x})$                          | 79 |
| 17.12.7 Průměrné skóre pro shluk $C_i$ : $s_i$  | 79 |
| 17.12.8 Průměrné skóre pro celé shlukování: $s$                                       | 79 |
| 17.13 Shlukování pomocí algoritmu DBSCAN  | 80 |
| 17.13.1 $\varepsilon$ okolí bodu ( $\varepsilon$ -neighborhood)                       | 80 |
| 17.13.2 Podmínka pro klíčový bod (core point)   | 80 |
| 17.14 Asociační pravidla  | 80 |
| 17.14.1 Podpora množiny položek $\mathcal{K}$   | 80 |
| 17.14.2 Spolehlivost (confidence)   | 80 |
| 17.14.3 Zdvih (lift)  | 80 |
| 17.14.4 Pokrytí (coverage)  | 80 |
| 17.14.5 Podmínky pro výběr pravidel   | 80 |

# 1 Obecné pojmy

## 1.1 Klasifikace a regrese (Classification and Regression)

- **Klasifikace (angl. Classification):** Problém, kdy vysvětlovaná proměnná  $Y$  (angl. target variable) může nabývat jen několik málo hodnot.
  - Příklady: určení, jestli pacient má/nemá nemoc či jaké písmeno je vyobrazeno na obrázku.
- **Regres (angl. Regression):** Problém, kdy vysvětlovaná proměnná  $Y$  (angl. target variable) může nabývat mnoha hodnot, že je rozumnější ji považovat za spojitou.
  - Příklady: Předpovídání ceny, teploty či věku.
- Oba typy patří do supervizovaného učení, kde se snažíme zjistit funkční vztah:

$$Y = f(X_1, X_2, \dots, X_p), \quad p \in \mathbb{N}.$$

## 1.2 Supervizované a nesupervizované učení

- **Supervizované učení (angl. Supervised Learning):** typ učení, kde máme známé hodnoty vysvětlované proměnné  $Y$  (angl. target variable), což je veličina, kterou se snažíme pomocí modelu predikovat resp. pochopit, na čem závisí. Tím "učitelem" jsou zde známé hodnoty  $Y$  v trénovacích datech.
  - Označováno taktéž jako **učení s učitelem**.
  - Příklady: Rozhodovací stromy (angl. Decision Trees), k-Nejbližších sousedů (angl. k-Nearest Neighbors), lineární regrese (angl. Linear Regression), hřebenová regrese (angl. Ridge Regression), logistická regrese (angl. Logistic Regression), náhodné lesy (angl. Random Forests), AdaBoost, XGBoost, Support Vector Machines (SVM).
- **Nesupervizované učení (angl. Unsupervised Learning):** nastává v situaci, kdy data nemáme nikterak označena. Tj. nemáme žádnou veličinu, kterou bychom u trénovacích dat znali a snažili se ji naučit predikovat. Cílem nesupervizovaného učení je porozumět struktuře dat pouze na základě nich samotných. To znamená bez nějakého vnějšího vodítka. Proto se nesupervizovanému učení také říká učení bez učitele.
  - Označováno taktéž jako **učení bez učitele**.
  - Příklady: Hierarchické shlukování (angl. Hierarchical Clustering), k-Means, DBSCAN (Density-Based Spatial Clustering), Asociační pravidla (Association Rules), PCA (Principal Component Analysis), Hidden Markov Models.

## 1.3 Nominální a ordinální příznaky

- **Nominální příznaky (angl. Nominal features):** Jsou kategorické příznaky, které nemají žádné přirozené uspořádání mezi kategoriemi. Kategorie jsou pouze odlišné "jmény" nebo "štítky" bez vztahu nadřazenosti či podřazenosti.
- Příklady:
  - Barva: {červená, modrá, zelená}
  - Jména: {Pavel, Ivan, ...}
  - Město: {Praha, Brno, Ostrava}
- **Ordinální příznaky (angl. Ordinal features):** Jsou kategorické příznaky, které mají přirozené uspořádání mezi kategoriemi. Lze říct, že jedna kategorie je "větší", "lepší" nebo "vyšší" než druhá, ale rozdíly mezi kategoriemi nemusí být stejně.
- Příklady:
  - Vzdělání: {základní < střední < vysokoškolské}
  - Velikost trička: {XS < S < M < L < XL}
  - Hodnocení: {špatné < průměrné < dobré < výborné}

## 1.4 Singulární a regulární matice

- **Regulární matice (angl. Invertible/Non-singular matrix):**
  - Čtvercová matice  $\mathbf{A} \in \mathbb{R}^{n \times n}$ , pro kterou existuje inverzní matice  $\mathbf{A}^{-1}$  taková, že  $\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = I$ .
  - Ekvivalentní podmínky pro regularitu:
    - \*  $\det(\mathbf{A}) \neq 0$
    - \* Sloupce (resp. řádky)  $\mathbf{A}$  jsou lineárně nezávislé
    - \* Hodnost  $\text{rank}(\mathbf{A}) = n$
    - \* Soustava  $\mathbf{Ax} = \mathbf{b}$  má pro každé  $\mathbf{b}$  právě jedno řešení
- **Singulární matice (angl. Singular matrix):**
  - Čtvercová matice  $\mathbf{A} \in \mathbb{R}^{n \times n}$ , pro kterou neexistuje inverzní matice.

- Ekvivalentní podmínky pro singularitu:
  - \*  $\det(\mathbf{A}) = 0$
  - \* Sloupce (resp. řádky)  $\mathbf{A}$  jsou lineárně závislé
  - \* Hodnost  $\text{rank}(\mathbf{A}) < n$
  - \* Existuje nenulový vektor  $\mathbf{v} \neq \mathbf{0}$  takový, že  $\mathbf{Av} = \mathbf{0}$
- **Souvislost s lineární regresí** (viz sekce 4):
  - Normální rovnice  $\mathbf{X}^T \mathbf{X} \mathbf{w} = \mathbf{X}^T \mathbf{Y}$  má jednoznačné řešení, pokud je matice  $\mathbf{X}^T \mathbf{X}$  regulární.
  - Pokud je  $\mathbf{X}^T \mathbf{X}$  singulární (lineárně závislé sloupce, kolinearita příznaků), nelze přímo vypočítat  $(\mathbf{X}^T \mathbf{X})^{-1}$ .
  - **Řešení:** SVD rozklad poskytuje stabilní výpočet pomocí Moorovy-Penroseovy pseudoinverze i pro singulární případy:  $\hat{\mathbf{w}} = \mathbf{V} \mathbf{\Sigma}^+ \mathbf{U}^T \mathbf{Y}$ .

## 1.5 Parametry vs. hyperparametry

- **Parametry (angl. Parameters):** Jsou hodnoty, které se model učí přímo z trénovacích dat během trénování. Parametry definují naučený model a používají se při predikci.
  - Příklady: Váhy  $\mathbf{w}$  v lineární regresi, pravděpodobnosti přechodů v HMM, váhy neuronové sítě.
  - Parametry se **optimalizují automaticky** během trénování (např. minimalizací ztrátové funkce).
- **Hyperparametry (angl. Hyperparameters):** Jsou hodnoty, které musí být nastaveny **před** zahájením trénování a řídí samotný proces učení nebo strukturu modelu. Model se je neučí z dat.
  - Příklady: `max_depth` u rozhodovacích stromů, počet sousedů  $k$  v kNN, learning rate u gradientního sestupu, regularizační parametr  $\lambda$  v hřebenové regresi.
  - Hyperparametry se **ladí ručně** nebo pomocí technik jako grid search nebo random search na validační množině.
- **Shrnutí:**
  - **Parametry:** Učí se z trénovacích dat (např.  $\mathbf{w}$  v  $Y = \mathbf{w}^T \mathbf{x}$ ).
  - **Hyperparametry:** Nastavují se před trénováním (např.  $k$  v kNN).

## 1.6 Parametrické vs. neparametrické modely

- **Parametrické modely (angl. Parametric models):** Předpokládají **pevnou funkční formu**. Počet parametrů je fixní (nezávisí na  $N$ , kde  $N$  je počet řádků dat).
  - Model si pamatuje jen pár čísel (váhy), trénovací data můžeme zahodit.
  - Příklad: Lineární regrese — předpokládá  $Y = \mathbf{w}^T \mathbf{x}$ . Pro  $p$  příznaků má  $p + 1$  vah. Natrénuješ na 1 000 000 bodech → pamatuješ jen  $p + 1$  čísel. Predikce = dosazení do vzorce.
  - Další příklady: Logistická regrese, naivní Bayesův klasifikátor, neuronové sítě (fixní architektura = fixní počet vah).
  - Výhody: Rychlá predikce, méně paměti, interpretovatelné.
- **Neparametrické modely (angl. Non-parametric models):** **Žádný předpoklad** o funkční formě. Složitost modelu roste s  $N$ , kde  $N$  je počet řádků dat.
  - Model si pamatuje celá trénovací data (nebo strukturu odvozenou z dat).
  - Příklad: kNN — žádný předpoklad, jestli to bude přímka, parabola, nebo cokoliv jiného. Musí mít uložené všechny trénovací body. Natrénuješ na 1 000 000 bodech → musíš pamatovat všech 1 000 000 bodů. Predikce = projet všechny body a najít  $k$  nejbližších.
  - Další příklady: Rozhodovací stromy (složitost stromu roste s daty), kernel regrese.
  - Nevýhody: Pomalé predikce (musíš projet data), více paměti.
- **Poznámka:** Název "neparametrický" je zavádějící — tyto modely mají parametry, ale jejich počet není fixní a roste s daty.

## 1.7 Přeučení vs. podučení (Overfitting vs. Underfitting)

- **Přeučení (angl. Overfitting):** Nastává, když se model **příliš přizpůsobí** trénovacím datům včetně náhodného šumu  $\epsilon$ , takže dobře funguje na trénovací množině, ale špatně generalizuje na nová (testovací) data.
  - **Příznaky:** Nízká chyba na trénovací množině, vysoká chyba na testovací množině.
  - **Příčiny:** Příliš složitý model, málo trénovacích dat, žádná regularizace.
  - **Příklady:** Rozhodovací strom s neomezenou hloubkou, kNN s  $k = 1$ , polynomiální regrese vysokého stupně.
  - **Řešení:** Regularizace (ridge, lasso), omezení složitosti modelu (`max_depth`), více dat, cross-validation, early stopping.
- **Podučení (angl. Underfitting):** Nastává, když je model **příliš jednoduchý** a nedokáže zachytit skutečný vztah mezi příznaky a vysvětlovanou proměnnou. Špatně funguje jak na trénovací, tak na testovací množině.
  - **Příznaky:** Vysoká chyba na trénovací i testovací množině.
  - **Příčiny:** Příliš jednoduchý model, nedostatek příznaků, příliš silná regularizace.
  - **Příklady:** Lineární regrese pro silně nelineární data, rozhodovací strom s `max_depth=1`, kNN s velmi velkým  $k$ .
  - **Řešení:** Složitější model, více příznaků, feature engineering, snížení regularizace.
- **Bias-Variance Trade-off:**
  - **Podučení:** Vysoký bias (model je systematicky nepřesný), nízká variance.
  - **Přeučení:** Nízký bias, vysoká variance (model je přecitlivělý na konkrétní trénovací data).
  - **Cíl:** Najít rovnováhu mezi biasem a variancí pomocí ladění hyperparametrů na validační množině.

## 2 Rozhodovací stromy (Decision Trees)

**Otzážka ke zkoušce 1 (Rozhodovací stromy):** Algoritmus ID3, kritéria pro větvení (entropie, gini, MSE), použití pro klasifikaci a regresi, hyperparametry rozhodovacích stromů.

### 2.1 Rozhodovací stromy: výhody a nevýhody

- Neparametrický model – počet parametrů (uzlů stromu) není fixní a závisí na datech. Strom si pamatuje pouze pravidla větvení a predikce v listech, ne samotná trénovací data.
- Výhody:
  - Nenáročnost na přípravu dat.
  - Jednoduché.
  - Srozumitelné.
  - Relativně rychlé učení.
  - Dobře interpretovatelné.
- Nevýhody:
  - Nerobustní, tj. drobná změna v trénovacích datech může znamenat zásadní změnu struktury výsledného stromu.
  - Většina implementací podporuje pouze binární stromy.
  - Snadno se přeučí (tj. overfitting).
  - Najít optimální strom hrubou silou je neproveditelné kvůli obrovskému počtu možných stromů.

## 2.2 Rozhodovací stromy: konstrukce

- Stromů existuje obrovské množství, tj. nemůžeme vyzkoušet všechny možnosti a vybrat tu nejlepší.
- Pro klasifikaci (angl. classification) používám **Entropii** (angl. Entropy) či **Gini index** (angl. Gini impurity), pro regresi (angl. regression) **MSE** (Mean Squared Error). Moderní implementace (CART) umí obojí.
- Protože nalezení optimálního stromu je NP-úplný problém, používají se v praxi **hladové algoritmy** (angl. Greedy Algorithms) (např. ID3, C4.5, C5 či CART), které strom budují rekurzivně výběrem příznaku, jenž v daném kroku nejlépe rozděluje data (maximalizuje informační zisk).
  - **ID3 (Iterative Dichotomiser 3):** Je určen primárně pro klasifikaci (pracuje s Entropií (angl. Entropy) či alternativně s Gini Index (angl. Gini impurity) pro diskrétní třídy). Vylepšené verze jsou C4.5 a C5.
  - **CART (Classification and Regression Trees):** Moderní algoritmus (používaný v `scikit-learn`). Pro klasifikaci typicky využívá **Gini index (angl. Gini impurity)** (lze i Entropii (angl. Entropy)), pro regresi (angl. regression) **MSE (Mean Squared Error)**. Vytváří binární stromy.

## 2.3 ID3 (Iterative Dichotomiser 3)

- Vytvořil vědec a výzkumník Ross Quinlan.

- **Pseudokód:**

- **Vstup:** Máme  $N$  řádkovou tabulkou s hodnotami pro binární promennou a  $p$  binárních příznaků (angl. features)  $X_1, X_2, \dots, X_p$ .
- **1. krok:** Spočítáme informační zisk pro každý příznak a vybereme (resp. sloupec), kde je informační zisk největší.
- **2. krok:** Vytvoříme uzel stromu. Jelikož jsou naše příznaky binární, strom se v tomto uzlu rozdělí na dvě větve (splní/nesplní podmínu).
- **Rekurze:** Data se rozdělí na dvě hromádky a celý postup opakujeme pro tyto menší hromádky, ale již bez příznaků, které jsme vybrali s největším informačním ziskem.
- **Konec:** Zastavovací kritérium rekurze je maximální hloubka, vyčerpání příznaků či když data jsou dokořane čistá (*Entropie* = 0), tj. všechny instance v uzlu patří do stejné třídy.

- **Entropie (angl. Entropy):**

- Entropie je míra neuspořádanosti množiny dat.
- Nechť je dána množina  $\mathcal{D}$  nul a jedniček (nebo i více hodnot) a chceme nějak změřit, jak moc je uspořáданá.
- Nechť  $p_0$  a  $p_1$  označují poměry počtu 0 resp. 1 v množině  $\mathcal{D}$  (tj.  $p_0 + p_1 = 1$ ).
- Taková míra by měla splňovat:
  1. Míra by měla být nezáporná.
  2. Pokud jsou v množině  $\mathcal{D}$  např. samé nuly (tj.  $p_0 = 1$ ), měla by být neuspořádanost nulová.
  3. Měla by být maximální, pokud jsou počty nul a jedniček stejně, tj. když  $p_0 = p_1 = 1/2$ .
  4. Měla by to být rostoucí funkce  $p_0$  na intervalu  $[0, 1/2]$  a klesající na intervalu  $[1/2, 1]$ .
- Taková funkce měřící neuspořádanost existuje a říká se jí entropie:

$$H(\mathcal{D}) = -p_0 \log p_0 - p_1 \log p_1 = -p_0 \log p_0 - (1 - p_0) \log(1 - p_0)$$

Ve vzorci se nejčastěji používá dvojkový logaritmus. V takovém případě se jednotce entropie říká **bit**. (např.  $H(\mathcal{D})$  se bude rovnat 0.81 a jednotce se říká 0.81 bitu, tj.  $H(\mathcal{D}) = 0.81 \text{ bitu}$ ).

- **Informační zisk (angl. Information Gain):**

- Informační zisk je číslo (metrika), které určuje, o kolik se snížila neuspořádanost (entropie) dat poté, co jsme je rozdělili podle určitého příznaku.

- Formálně:

$$IG(\mathcal{D}, X_i) = H(\mathcal{D}) - t_0 H(\mathcal{D}_0) - t_1 H(\mathcal{D}_1), \quad i \in \{1, \dots, p\}$$

kde pro  $v \in \{0, 1\}$  je  $\mathcal{D}_v = \{\mathbf{x} \in \mathcal{D} \mid x_i = v\}$  a  $t_v$  je podíl počtu prvků v  $\mathcal{D}_v$  a  $\mathcal{D}$ , neboli  $t_v = \frac{|\mathcal{D}_v|}{|\mathcal{D}|}$ . V tomto vzorci  $X_i$  označuje  $i$ -tý sloupec (resp. příznak) matice  $\mathbf{X}$ , což je matice dat.

- **Příklad výpočtu entropie a informačního zisku:**

Mějme následující trénovací data s vysvětlovanou proměnnou  $Y$  (vstal/nevstal) a dvěma binárními příznaky:

| <b>id</b> | <b>teplota &gt; 39°C (<math>X_1</math>)</b> | <b>bolest hlavy (<math>X_2</math>)</b> | <b>vstal(a)? (<math>Y</math>)</b> |
|-----------|---|--|-----------------------------------|
| 1         | ano   | ano                                    | ne                                |
| 2         | ne  | ne                                     | ano                               |
| 3         | ne  | ne                                     | ano                               |
| 4         | ano   | ano                                    | ne                                |
| 5         | ano   | ano                                    | ne                                |
| 6         | ne  | ano                                    | ano                               |

Tabulka 1: Trénovací data pro výpočet entropie a informačního zisku.

**Poznámka k použití logaritmu:** Ve vzorci pro entropii používáme logaritmus z praktických důvodů. Při násobení pravděpodobností (čísel z intervalu  $[0, 1]$ ) bychom rychle dostávali extrémně malá čísla. To je problém, protože počítače mají omezenou přesnost reprezentace čísel s plovoucí řádovou čárkou — při příliš malých hodnotách dochází k tzv. **numerickému podtečení (angl. underflow)**, kdy je číslo zaokrouhleno na nulu a ztrácíme veškerou informaci. Logaritmus má tu vlastnost, že převádí násobení na sčítání:  $\log(a \cdot b) = \log a + \log b$ , čímž se tomuto problému vyhneme — místo součinu velmi malých čísel sčítáme záporná čísla běžných velikostí.

**Krok 1: Výpočet entropie celého datasetu  $H(\mathcal{D})$**

V datasetu máme 6 záznamů: 3 s  $Y = 1$  (vstal) a 3 s  $Y = 0$  (nevstal). Tedy:

$$p_1 = \frac{3}{6} = \frac{1}{2}, \quad p_0 = \frac{3}{6} = \frac{1}{2}$$

Dosadíme do vzorce pro entropii:

$$\begin{aligned} H(\mathcal{D}) &= -p_0 \log_2 p_0 - p_1 \log_2 p_1 \\ &= -\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2} \\ &= -\frac{1}{2} \cdot (-1) - \frac{1}{2} \cdot (-1) \\ &= \frac{1}{2} + \frac{1}{2} = 1 \text{ bit} \end{aligned}$$

Entropie 1 bit odpovídá maximální neuspořádanosti pro binární proměnnou (obě třídy jsou stejně zastoupeny).

**Krok 2: Výpočet informačního zisku pro příznak  $X_1$  (teplota > 39°C)**

Rozdělíme data podle příznaku  $X_1$ :

- $\mathcal{D}_1 = \{\text{záznamy kde } X_1 = 1 \text{ (ano)}\} = \{1, 4, 5\} \Rightarrow 3 \text{ záznamy, z toho } 0 \text{ vstal, } 3 \text{ nevstal}$
- $\mathcal{D}_0 = \{\text{záznamy kde } X_1 = 0 \text{ (ne)}\} = \{2, 3, 6\} \Rightarrow 3 \text{ záznamy, z toho } 3 \text{ vstal, } 0 \text{ nevstal}$

Váhy podmnožin:  $t_1 = \frac{3}{6} = \frac{1}{2}$ ,  $t_0 = \frac{3}{6} = \frac{1}{2}$

Entropie podmnožin:

- Pro  $\mathcal{D}_1$ :  $p_1 = 0, p_0 = 1 \Rightarrow H(\mathcal{D}_1) = -0 \cdot \log_2 0 - 1 \cdot \log_2 1 = 0 \text{ bit}$  (konvence:  $0 \cdot \log_2 0 = 0$ )
- Pro  $\mathcal{D}_0$ :  $p_1 = 1, p_0 = 0 \Rightarrow H(\mathcal{D}_0) = -1 \cdot \log_2 1 - 0 \cdot \log_2 0 = 0 \text{ bit}$

Informační zisk pro  $X_1$ :

$$\begin{aligned} IG(\mathcal{D}, X_1) &= H(\mathcal{D}) - t_1 H(\mathcal{D}_1) - t_0 H(\mathcal{D}_0) \\ &= 1 - \frac{1}{2} \cdot 0 - \frac{1}{2} \cdot 0 \\ &= 1 \text{ bit} \end{aligned}$$

### Krok 3: Výpočet informačního zisku pro příznak $X_2$ (bolest hlavy)

Rozdělíme data podle příznaku  $X_2$ :

- $\mathcal{D}_1 = \{\text{záznamy kde } X_2 = 1\} = \{1, 4, 5, 6\} \Rightarrow 4 \text{ záznamy, z toho 1 vstal, 3 nevstal}$
- $\mathcal{D}_0 = \{\text{záznamy kde } X_2 = 0\} = \{2, 3\} \Rightarrow 2 \text{ záznamy, z toho 2 vstal, 0 nevstal}$

Váhy:  $t_1 = \frac{4}{6} = \frac{2}{3}$ ,  $t_0 = \frac{2}{6} = \frac{1}{3}$

Entropie podmnožin:

- Pro  $\mathcal{D}_1$ :  $p_1 = \frac{1}{4}$ ,  $p_0 = \frac{3}{4} \Rightarrow H(\mathcal{D}_1) = -\frac{1}{4} \log_2 \frac{1}{4} - \frac{3}{4} \log_2 \frac{3}{4} \approx 0.81 \text{ bit}$
- Pro  $\mathcal{D}_0$ :  $p_1 = 1$ ,  $p_0 = 0 \Rightarrow H(\mathcal{D}_0) = 0 \text{ bit}$

Informační zisk pro  $X_2$ :

$$\begin{aligned} IG(\mathcal{D}, X_2) &= H(\mathcal{D}) - t_1 H(\mathcal{D}_1) - t_0 H(\mathcal{D}_0) \\ &= 1 - \frac{2}{3} \cdot 0.81 - \frac{1}{3} \cdot 0 \\ &\approx 1 - 0.54 = 0.46 \text{ bit} \end{aligned}$$

**Závěr:** Příznak  $X_1$  (teplota  $> 39^\circ\text{C}$ ) má informační zisk 1 bit, zatímco  $X_2$  (bolest hlavy) má pouze 0.46 bit. Algoritmus ID3 by tedy v prvním kroku vybral příznak  $X_1$  pro větvení, protože má největší informační zisk — po rozdělení podle tohoto příznaku jsou obě podmnožiny zcela homogenní (entropie 0).

- **Gini Index (angl. Gini impurity):**

- Gini Index je alternativní metrika k entropii. Má podobné vlastnosti jako Entropie.
- Je to jakási míra toho, že nově přidaný prvek bude špatně klasifikován.
- Jinak ale vše funguje stejně, jen se nahradí  $H(\mathcal{D})$  výrazem  $GI(\mathcal{D})$ .
- Pro množinu  $\mathcal{D}$  s  $k$  různými hodnotami Gini Index formálně definujeme jako:

$$GI(\mathcal{D}) = 1 - \sum_{i=0}^{k-1} p_i^2 = \sum_{i=0}^{k-1} p_i(1 - p_i)$$

kde  $p_i$  je relativní četnost tj.

$$p_i = \frac{\text{počet vzorků třídy } i}{\text{celkový počet vzorků}}$$

## 2.4 Regrese & CART (Classification and Regression Trees)

- Pro práci se spojitou vysvětlovanou proměnnou.
- **Predikce hodnoty vysvětlované proměnné:**
  - Obvykle bereme průměr (resp. lze i medián či modus) hodnot z listu, někdy může být zavádějící, stačí když se tam dostane jedna extrémní hodnota (angl. outlier).
- **CART (Classification and Regression Trees) pro Regresi:**
  - Místo entropie (popř. Gini Index) použijeme MSE (Mean Squared Error), což je skoro stejná veličina jako výběrový rozptyl (angl. Sample Variance).

- MSE (Mean Squared Error) definujeme předpisem:

$$MSE(Y) = \frac{1}{N} \sum_{i=0}^{N-1} (Y_i - \bar{Y})^2$$

- MSE lze nahradit i MAE (Mean Absolute Error), i když je to méně obvyklé. MAE definujeme takto:

$$MAE(Y) = \frac{1}{N} \sum_{i=0}^{N-1} |Y_i - \bar{Y}|$$

MAE není derivovatelné v bodě  $Y_i = \bar{Y}$ , což komplikuje gradient-based optimalizaci. Proto se v praxi častěji používá MSE.

- Funguje stejně jako ID3 — rozdělí  $\mathcal{D}$  na  $\mathcal{D}_L$  a  $\mathcal{D}_R$ , ale místo entropie:

$$IG(\mathcal{D}, X_i) = H(\mathcal{D}) - t_L H(\mathcal{D}_L) - t_R H(\mathcal{D}_R), i \in \{1, \dots, p\}$$

používá:

$$IG(\mathcal{D}, X_i) = MSE(\mathcal{D}) - t_L MSE(\mathcal{D}_L) - t_R MSE(\mathcal{D}_R), i \in \{1, \dots, p\}$$

kde  $t_L = \frac{|\mathcal{D}_L|}{|\mathcal{D}|}$  a  $t_R = \frac{|\mathcal{D}_R|}{|\mathcal{D}|}$ . V tomto vzorci  $X_i$  označuje  $i$ -tý sloupec (resp. příznak) matice  $\mathbf{X}$ , což je matice dat.

## 2.5 Hyperparametry rozhodovacích stromů

- **Ladění rozhodovacích stromů:**

- `max_depth`: Hloubka stromu.
- `criterion`: Funkce kvality splitu (gini, entropie, ...).
- `min_samples_split`: Minimální počet prvků potřebných pro rozdělení vnitřního uzlu.
- `min_samples_leaf`: Minimální počet vzorků, které musí obsahovat každý list (koncový uzel).

- **Nebinární příznaky (one-hot-encoding):**

- Jeden kategorický příznak s  $n$  různými hodnotami se nahradí  $n - 1$  binárními dummy proměnnými.
- Mohli bychom vytvořit i sloupec pro "spadl", ale byl by redundantní, protože informace je již obsažena v předchozích dvou sloupcích.
- Příklad: příznak "stav"s hodnotami {vstal, nevstal, spadl} se zakóduje pomocí dvou binárních příznaků následovně:

| Původní příznak ( $X$ ) | $X_{vstal}$ | $X_{nevstal}$ |
|-------------------------|-------------|---------------|
| vstal                   | 1           | 0             |
| nevstal                 | 0           | 1             |
| spadl                   | 0           | 0             |

Tabulka 2: Dummy encoding: 3 kategorie zakódovány pomocí 2 binárních příznaků ( $n - 1$ ).

- Nevýhody:

- Zvyšuje počet příznaků (dimenze datasetu).
- Není vhodná pro ordinální příznaky (uspořádané jako základní < střední < univerzitní), je vhodnější pro nominální (mezi kategoriemi není žádný vztah).

- **Spojité příznaky:**

- Moderní algoritmy (CART, C4.5, C5) zpracovávají spojité příznaky automaticky pomocí porovnání s prahem (např.  $X < 5.2$ ).

- **Hledání optimálního prahu:** Algoritmus seřadí hodnoty příznaku a pro každý možný prah spočítá informační zisk. Vybere práh s největším ziskem.
- Stejný příznak lze použít opakováně s různými prahy v různých částech stromu (např. "věk < 30", později "věk < 15").
- **Poznámka:** Původní ID3 vyžaduje diskretizaci spojitých příznaků. C4.5, C5 a CART je zpracovávají automaticky.

## 2.6 Zajímavá videa na YouTube

- [Entropy \(for data science\) Clearly Explained!!! \(StatQuest with Josh Starmer\)](#) [16 min]
- [Shannon Entropy and Information Gain \(Serrano.Academy\)](#) [21 min]

## 3 kNN (k-Nearest Neighbor)

**Otázka ke zkoušce 2 (Metoda nejbližších sousedů: kNN):** Popis metody, hyperparametrů a jejího použití pro klasifikaci a regresi. Pojem metrika a normalizace dat.

### 3.1 kNN: základní popis

- **kNN (k-Nearest Neighbors):** Metoda nejbližších sousedů je algoritmus pro supervizované učení, který funguje jak pro klasifikaci, tak pro regresi.
- **Princip metody:**
  - Má se predikovat hodnota vysvětlované proměnné pro datový bod  $\mathbf{x} \in \mathbb{R}^p, p \in \mathbb{N}$ .
  - V trénovacích datech najdeme  $k$  (zadaný hyperparametr) nejbližších bodů k  $\mathbf{x}$ .
  - Vzdálenost měříme pomocí **metriky** — funkce, pro kterou platí:
    1. pozitivní definitnost
    2. symetrie
    3. trojúhelníková nerovnost
  - Příklad metriky je Euklidovská vzdálenost.
  - Predikci založíme na známých hodnotách vysvětlované proměnné pro těchto  $k$  bodů:
    - \* **Pro regresi:** Bereme průměr (popřípadě medián) hodnot.
    - \* **Pro klasifikaci:** Bereme nejčastější hodnotu (modus).
- **kNN učení vs. predikování:**
  - Pro většinu metod pro supervizované učení platí, že řádově více je náročné učení modelu (trénování, vytváření stromu) než výpočet predikcí.
  - U kNN je tomu obráceně — **učení neprobíhá:** trénovací data jsou sama o sobě naučeným modelem, náročná je predikce — hledání nejbližších sousedů (lze zrychlit indexací).

### 3.2 kNN: hyperparametry

- **Číslo  $k$  určující počet nejbližších sousedů:**
  - Čím větší počet sousedů ( $k$ ), tím menší šance na přeucení (pokud bereme všechny — děláme průměr ze všech, pokud bereme jeden — blížíme se konkrétním trénovacím datům).
- **Použitá vzdálenost (metrika):**

- Nejoblíbenější volbou jsou tzv. **k-normy** (také  $L_k$  případně Minkovského  $k$ -metriky):

$$\|\mathbf{x} - \mathbf{y}\|_k = d_k(\mathbf{x}, \mathbf{y}) = \sqrt[k]{\sum_{i=1}^p |x_i - y_i|^k}$$

kde  $k \in \mathbb{N}$ .

- Pro  $k = 2$  dostáváme **Euklidovskou vzdálenost**:

$$\|\mathbf{x} - \mathbf{y}\|_2 = d_2(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^p (x_i - y_i)^2}$$

- Pro  $k = 1$  dostáváme **Manhattanskou vzdálenost**:

$$\|\mathbf{x} - \mathbf{y}\|_1 = d_1(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^p |x_i - y_i|$$

- **Váhy nejbližších sousedů:**

- Co, když jeden parametr bude v mm a druhý v m?
- Co, když jeden je v počtu pokojů a druhý parametr v úhlopříčce televize?
- Je potřeba **normalizace** do intervalu  $[0, 1]$  nebo **standardizace**.

### 3.3 Normalizace dat

- **Min-max normalizace (do intervalu  $[0, 1]$ ):**

- Pro daný příznak najdeme jeho minimální a maximální hodnotu v trénovacích datech:  $\min_x, \max_x$ .
- Pak hodnotu  $x_i$  tohoto příznaku pro  $i$ -tý datový bod nahradíme:

$$x_i \leftarrow \frac{x_i - \min_x}{\max_x - \min_x}$$

- Tím docílíme toho, že všechny hodnoty budou z intervalu  $[0, 1]$ .

- **Nevýhody:**

- \* Této metodě škodí outliers (odlehlé hodnoty).
- \* Zahazujeme informaci o vztahu mezi příznaky (např. jeden je 2x co druhý).
- \* Obecně složité téma, neexistuje univerzální správný přístup.

- V **scikit-learn** je pod názvem **MinMaxScaler**.

- **Standardizace (z-score normalizace):**

- Alternativa normalizace.
- Pro každý příznak nalezneme jeho výběrový průměr  $\bar{x}$  a výběrový rozptyl  $s_x^2$  a každý  $i$ -tý bod nahradíme:

$$x_i \leftarrow \frac{x_i - \bar{x}}{\sqrt{s_x^2}}$$

kde  $\bar{x}$  je výběrový průměr a  $s_x^2$  je výběrový rozptyl.

- Také citlivé na outliers, ale méně nežli **Min-max normalizace**.
- Převádí na škálu s  $\bar{x} = 0$  (výběrový průměr) a  $\sqrt{s_x^2} = 1$  (výběrová směrodatná odchylka).
- V **scikit-learn** je pod názvem **StandardScaler**.

- **Další metody škálování v scikit-learn:**

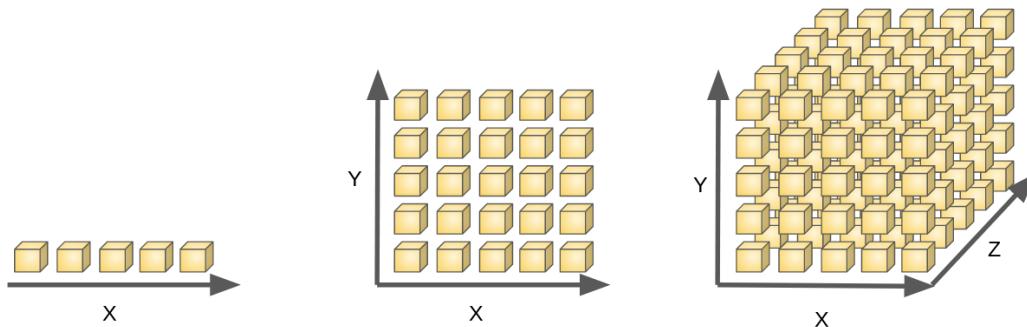
- **RobustScaler** - používá medián a mezikvartilové rozpětí (IQR), odolnější vůči outliers než standardizace.
- **MaxAbsScaler** - škáluje podle maximální absolutní hodnoty do intervalu  $[-1, 1]$ .
- A další (**Normalizer**, **QuantileTransformer**, **PowerTransformer**, atd.)

### 3.4 kNN a nominální příznaky

- kNN se bez použití speciálních metrik neumí dobře vypořádat s nominálními příznaky (neuspořádanými).
- Můžeme použít **one-hot encoding** a **dummy příznaky** (viz. sekce Rozhodovací stromy).

### 3.5 Prokletí dimenzionality (The Curse of Dimensionality)

- **Prokletí dimenzionality (angl. The curse of dimensionality):** Je pojem, který odkazuje na některé problémy objevující se v případě vysokého počtu příznaků, kdy jsou datové body prvky mnohadimenzionálního prostoru.
- Proto je potřeba někdy provést **redukci dimenzionality**, např. výběr příznaků (**Feature Selection**).
- **S kNN jsou spojeny zejména 2 efekty způsobené vysokou dimenzí:**
  - Data se zvyšováním dimenzí řídnou a navzájem se vzdalují. Pro zachování stejné hustoty pro vyšší dimenze by bylo nutné řádově navýšit počet datových bodů, což většinou není možné.
  - S rostoucí dimenzí se pro klasické metriky zmenšují rozdíly mezi vzdálenými a blízkými body.
- Ve velké dimenzi jsou všichni sousedi daleko, takže jejich zprůměrováním nedostaneme dobrý odhad.
- Příklad 1:



Obrázek 1: Prokletí dimenzionality (řídnutí bodů)

Obrázek 1 ilustruje **řídnutí dat** (angl. Data Sparsity) s rostoucí dimenzí. Stejný počet bodů je v 1D hustě vedle sebe, v 2D se rozloží do plochy a jsou řidší, v 3D se rozloží do prostoru a vzdálenosti mezi nimi dramaticky rostou. Data se zvyšováním dimenze řídnou a navzájem se vzdalují — hustota trénovacích bodů s rostoucí dimenzí klesá.

- Příklad 2: Mějme  $d$ -dimenzionální jednotkovou hyperkrychli:

$$[0, 1]^d, d \in \mathbb{N}$$

s 1000 náhodně rozmištěnými body. Chceme najít oblast, která obsahuje v průměru 10 bodů, tj. pokrývá 1% celkového objemu. **Otzáka:** Jak velkou hyperkrychli o straně  $a$  potřebujeme, aby obsahovala 1% objemu?

V dimenzi  $d$  musí platit:  $a^d = 0.01$ , tedy  $a = \sqrt[d]{0.01}$ .

**Konkrétní výpočet:**

- **1D:**  $a = 0.01$  — potřebujeme 1% délky.
- **2D:**  $a = \sqrt{0.01} = 0.1$  — potřebujeme 10% na každou stranu.
- **3D:**  $a = \sqrt[3]{0.01} \approx 0.215$  — potřebujeme 21.5% na každou stranu.
- **5D:**  $a = \sqrt[5]{0.01} \approx 0.398$  — potřebujeme skoro 40% na každou stranu!
- **10D:**  $a = \sqrt[10]{0.01} \approx 0.63$  — potřebujeme 63% na každou stranu.

- **50D:**  $a = \sqrt[50]{0.01} \approx 0.91$  — potřebujeme 91% na každou stranu!
- **999D:**  $a = \sqrt[999]{0.01} \approx 0.9954$  — potřebujeme 99.54% na každou stranu — téměř celý prostor!

**Co to znamená?** Ačkoliv ve všech dimenzích pokrýváme stejné procento **objemu** (1%), musíme v každé dimenzi prohledat stále větší a větší podíl **rozsahu prostoru**. V 5D je to již skoro 40%, v 50D dokonce 91%, a v 999D se velmi blížíme ke 100%!

**Důsledek pro kNN:** Čím ve větší dimenzi hledáme nejbližší sousedy, tím větší prostor musíme prohledat, abychom našli dostatečný počet bodů. To znamená, že nejbližší sousedi jsou ve skutečnosti velmi daleko a jejich zprůměrováním nedostaneme dobrý odhad.

### 3.6 Zajímavá videa na YouTube

- What is the K-Nearest Neighbor (KNN) Algorithm? (IBM Technology) (vhodné pro prvotní myšlenku) [8 min]
- StatQuest: K-nearest neighbors, Clearly Explained (StatQuest with Josh Starmer) (prvotní myšlenka) [5 min]
- kNN.3 Voronoi cells and decision boundary (Victor Lavrenko) [3 min]
- kNN.4 Sensitivity to outliers (Victor Lavrenko) [2 min]
- kNN.5 Nearest-neighbor classification algorithm (Victor Lavrenko) [2 min]
- kNN.6 MNIST digit recognition (Victor Lavrenko) [4 min]
- kNN.7 Nearest-neighbor regression algorithm (Victor Lavrenko) [1 min]
- kNN.8 Nearest-neighbor regression example (Victor Lavrenko) [4 min]
- kNN.9 Number of nearest neighbors to use (Victor Lavrenko) [3 min]
- kNN.10 Similarity / distance measures (Victor Lavrenko) [5 min]
- kNN.11 Breaking ties between nearest neighbors (Victor Lavrenko) [3 min]
- kNN.12 Parzen windows, kernels and SVM (Victor Lavrenko) (nad rámec ML1, spadá do ML2) [13 min]
- kNN.13 Pros and cons of nearest-neighbor methods (Victor Lavrenko) [2 min]
- kNN.14 Computational complexity of finding nearest-neighbors (Victor Lavrenko) [3 min]
- kNN.16 Locality sensitive hashing (LSH) (Victor Lavrenko) (rozšíření nad rámec ML1) [7 min]
- kNN.17 Inverted index (Victor Lavrenko) (rozšíření nad rámec ML1) [4 min]

## 4 Lineární regrese (Linear Regression)

**Otzáka ke zkoušce 3 (Lineární regrese, metoda nejmenších čtverců):** Model lineární regrese, predikce, maticový zápis trénovací množiny. Metoda nejmenších čtverců: normální rovnice, řešení.

### 4.1 Metoda nejmenších čtverců (Ordinary Least Squares)

- V modelu lineární regrese **předpokládáme lineární závislost vysvětlované proměnné na hodnotách příznaků**, tedy:

$$Y = w_0 + w_1 x_1 + w_2 x_2 + \dots + w_p x_p + \varepsilon, \quad p \in \mathbb{N}$$

kde  $w_i, i \in \{0, 1, 2, \dots, p\}$  jsou nějaké neznámé koeficienty, které hledáme.  $\varepsilon$  je náhodná veličina s  $E(\varepsilon) = 0$ . Koeficient  $w_0$  se nazývá **intercept** a odpovídá (očekávané) výchozí hodnotě  $Y$  při nulových příznacích.

Pokud si označíme vektor příznaků jako  $\mathbf{x} = (1, x_1, \dots, x_p)^T$  a vektor váh jako  $\mathbf{w} = (w_0, w_1, \dots, w_p)^T$ , potom můžeme zapisovat kompaktněji:

$$Y = \mathbf{w}^T \mathbf{x} + \varepsilon$$

- Měření chyby predikce pomocí ztrátové funkce (angl. loss function)
  - Naším cílem je najít takovou hodnotu  $\mathbf{w}$ , aby chyba modelu byla co nejmenší.
  - Chybu modelu nejčastěji měříme pomocí nějaké nezáporné funkce  $L : \mathbb{R}^2 \rightarrow \mathbb{R}$ , nazývané ztrátová funkce (angl. loss function), kterou aplikujeme na skutečnou hodnotu proměnné  $Y$  a odpovídající predikci  $\hat{Y}$ .
  - Obvyklou volbou v případě spojité vysvětlované veličiny bývá kvadratická ztrátová funkce,

$$L(Y, \hat{Y}) = (Y - \hat{Y})^2$$

- Při trénování minimalizujeme **residuální součet čtverců** (angl. Residual Sum of Squares), který značíme  $RSS(\mathbf{w})$  a definujeme předpisem:

$$RSS(\mathbf{w}) = \sum_{i=1}^N L(Y_i, \mathbf{w}^T \mathbf{x}_i) = \sum_{i=1}^N (Y_i - \mathbf{w}^T \mathbf{x}_i)^2 = \|\mathbf{Y} - \mathbf{X}\mathbf{w}\|^2$$

kde  $N$  reprezentuje počet řádků v datasetu. Minimalizací tohoto výrazu získáme odhad  $\hat{\mathbf{w}}$ .

Toto chceme minimalizovat. Výraz si proto vhodně upravíme:

$$\begin{aligned} RSS(\mathbf{w}) &= \sum_{i=1}^N (Y_i - \mathbf{w}^T \mathbf{x}_i)^2 \\ &= (\mathbf{Y} - \mathbf{X}\mathbf{w})^T (\mathbf{Y} - \mathbf{X}\mathbf{w}) \\ &= \mathbf{Y}^T \mathbf{Y} - 2\mathbf{Y}^T \mathbf{X}\mathbf{w} + \mathbf{w}^T \mathbf{X}^T \mathbf{X}\mathbf{w} \end{aligned}$$

Hledáme minimum, proto sestrojíme **gradient**  $RSS(\mathbf{w})$ :

$$\begin{aligned} \nabla RSS(\mathbf{w}) &= \frac{\partial RSS(\mathbf{w})}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}} (\mathbf{Y}^T \mathbf{Y} - 2\mathbf{Y}^T \mathbf{X}\mathbf{w} + \mathbf{w}^T \mathbf{X}^T \mathbf{X}\mathbf{w}) \\ &= -2\mathbf{X}^T \mathbf{Y} + 2\mathbf{X}^T \mathbf{X}\mathbf{w} \end{aligned}$$

Potom **Hessova matice** (angl. Hessian Matrix)  $RSS(\mathbf{w})$  je:

$$\begin{aligned} \nabla^2 RSS(\mathbf{w}) &= \frac{\partial}{\partial \mathbf{w}} (\nabla RSS(\mathbf{w})) = \frac{\partial}{\partial \mathbf{w}} (-2\mathbf{X}^T \mathbf{Y} + 2\mathbf{X}^T \mathbf{X}\mathbf{w}) \\ &= \frac{\partial}{\partial \mathbf{w}} (-2\mathbf{X}^T \mathbf{Y}) + \frac{\partial}{\partial \mathbf{w}} (2\mathbf{X}^T \mathbf{X}\mathbf{w}) \\ &= 2\mathbf{X}^T \mathbf{X} \end{aligned}$$

Nyní musíme z definice dokázat, že Hessova matice je **pozitivně definitní**, abychom dokázali, že v řešení normální rovnice je lokální minimum. **Předpokládejme, že  $\mathbf{X}^T \mathbf{X}$  je regulární matice** (nebo ekvivalentně, že sloupce matice  $\mathbf{X}$  jsou lineárně nezávislé). Pro libovolný vektor  $\mathbf{v} \in \mathbb{R}^{p+1}$  platí:

$$\mathbf{v}^T (2\mathbf{X}^T \mathbf{X}) \mathbf{v} = 2\mathbf{v}^T (\mathbf{X}^T \mathbf{X}) \mathbf{v} = 2(\mathbf{X}\mathbf{v})^T (\mathbf{X}\mathbf{v}) = 2\|\mathbf{X}\mathbf{v}\|^2 > 0$$

Z lineární nezávislosti sloupců  $\mathbf{X}$  plyne, že pro  $(\forall \mathbf{v} \neq \mathbf{0})(\mathbf{X}\mathbf{v} \neq \mathbf{0})$ , a tedy:

$$2\|\mathbf{X}\mathbf{v}\|^2 > 0 \quad \text{pro } \forall \mathbf{v} \neq \mathbf{0}$$

Hessova matice je tedy pozitivně definitní, což znamená, že řešení normální rovnice  $\hat{\mathbf{w}}$  je bodem ostrého lokálního minima.

Nyní, když jsme dokázali, že v řešení normální rovnice nastává lokální minimum, můžeme z podmínky  $\nabla RSS(\mathbf{w}) = 0$  vyjádřit  $\mathbf{w}$ , čímž získáme odhad  $\hat{\mathbf{w}}$ :

$$\nabla RSS(\mathbf{w}) = 0 \iff -2\mathbf{X}^T \mathbf{Y} + 2\mathbf{X}^T \mathbf{X}\mathbf{w} = 0 \iff \mathbf{X}^T \mathbf{X}\mathbf{w} = \mathbf{X}^T \mathbf{Y} \iff \mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}$$

Odvozený výraz:

$$\hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}$$

nazýváme **řešením normální rovnice**.

## 4.2 Řešení normální rovnice pomocí SVD rozkladu

- Pro standardní výpočet  $\hat{\mathbf{w}}$  přes normální rovnici musíme vypočítat  $(\mathbf{X}^T \mathbf{X})^{-1}$ , což nelze pro singulární matice. Tato komplikace se řeší pomocí SVD rozkladu, který lze spočítat pro každou matici libovolných rozměrů.
- **SVD rozklad** (angl. Singular Value Decomposition) každé matice  $\mathbf{X} \in \mathbb{R}^{N \times (p+1)}$  má tvar:

$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$$

kde  $\mathbf{U} \in \mathbb{R}^{N \times N}$  a  $\mathbf{V} \in \mathbb{R}^{(p+1) \times (p+1)}$  jsou ortogonální matice ( $\mathbf{U}^T \mathbf{U} = I$ ,  $\mathbf{V}^T \mathbf{V} = I$ ) a  $\Sigma \in \mathbb{R}^{N \times (p+1)}$  je diagonální matice se singulárními hodnotami  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_{p+1} \geq 0$  na diagonále.

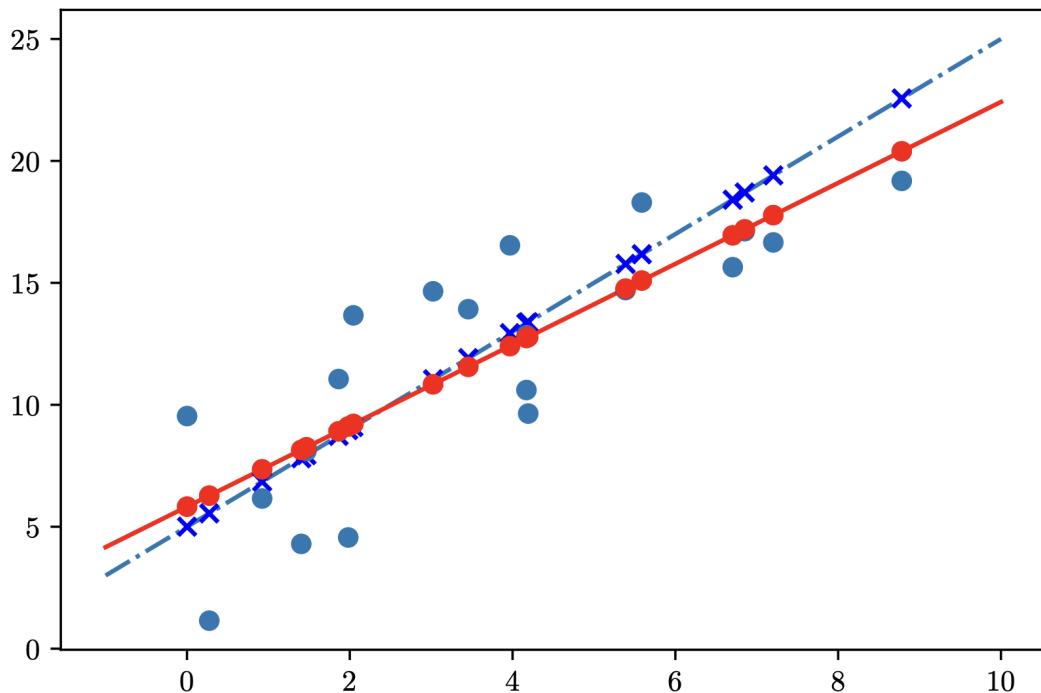
- $\hat{\mathbf{w}}$  lze proto stabilněji a univerzálněji spočítat pomocí SVD. Dosadíme-li  $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$  do řešení normální rovnice:

$$\begin{aligned}\hat{\mathbf{w}} &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y} \\ &= ((\mathbf{U}\Sigma\mathbf{V}^T)^T (\mathbf{U}\Sigma\mathbf{V}^T))^{-1} (\mathbf{U}\Sigma\mathbf{V}^T)^T \mathbf{Y} \\ &= (\mathbf{V}\Sigma^T \mathbf{U}^T \mathbf{U}\Sigma\mathbf{V}^T)^{-1} \mathbf{V}\Sigma^T \mathbf{U}^T \mathbf{Y} \\ &= (\mathbf{V}\Sigma^T \mathbf{V}^T)^{-1} \mathbf{V}\Sigma^T \mathbf{U}^T \mathbf{Y} \quad (\text{využili jsme } \mathbf{U}^T \mathbf{U} = I) \\ &= \mathbf{V}(\Sigma^T \Sigma)^{-1} \mathbf{V}^T \mathbf{V}\Sigma^T \mathbf{U}^T \mathbf{Y} \\ &= \mathbf{V}(\Sigma^T \Sigma)^{-1} \Sigma^T \mathbf{U}^T \mathbf{Y} \quad (\text{využili jsme } \mathbf{V}^T \mathbf{V} = I) \\ &= \mathbf{V}\Sigma^+ \mathbf{U}^T \mathbf{Y}\end{aligned}$$

kde  $\Sigma^+ = (\Sigma^T \Sigma)^{-1} \Sigma^T$  je **Moorova-Penroseova pseudoinverze** (angl. Moore-Penrose pseudoinverse) matice  $\Sigma$ .

- **Implementace v praxi:** Knihovna `scikit-learn` (`LinearRegression()`) používá pro výpočet  $\hat{\mathbf{w}}$  numericky stabilní SVD rozklad prostřednictvím funkce `dgesvd` z knihovny LAPACK, což zajišťuje korektní řešení i pro singulární matice.

## 4.3 Predikce lineární regrese



Obrázek 2: Vizualizace modelu lineární regrese

- Na obrázku 2: **modré body** jsou trénovací data  $(x_i, Y_i)$ , **červené body** jsou predikce  $\hat{Y}_i$ , **modré křížky** jsou střední hodnoty  $(x_i, E(Y_i))$ , **modrá čerchovaná čára** je skutečná přímka  $y = \mathbf{w}^T \mathbf{x}$  (neznámá), a **červená čára** je naše odhadnutá přímka  $\hat{y} = \hat{\mathbf{w}}^T \mathbf{x}$ . Nejlepší odhad by byl, kdyby se červená přímka dostala na modrou čerchovanou čáru.

#### 4.4 Závěrečné poznámky

- Lineární regrese je ve vztahu ke problémům dimenziality (narozdíl od kNN) poměrně **rezistentní**. Důvodem je, že se jedná o parametrickou metodu – v ideálním případě nám tedy pro  $p$  příznaků  $+1$  intercept může k určení přesného modelu stačit přesně  $p + 1$  bodů trénovací množiny.
- **Problémy nastávají**, když jsou v důsledku malé trénovací množiny nebo špatných příznaků, které jsou např. silně korelované, sloupce matice  $\mathbf{X}$  (skoro) lineárně závislé:
  - **Numerická nestabilita:** Přímý výpočet pomocí  $(\mathbf{X}^T \mathbf{X})^{-1}$  je numericky nestabilní nebo matice není invertibilní. Tento problém řeší SVD rozklad, který poskytuje stabilní výpočet i pro singulární nebo špatně podmíněné matice.
  - **Přeučení modelu:** I když SVD zvládne spočítat řešení, výsledný model může trpět přeučením (angl. overfitting), tj. příliš se přizpůsobí trénovací množině včetně náhodné chyby  $\varepsilon$  a nebude schopen dobře predikovat nové body. Řešením je použití regularizace (např. hřebenová regrese).
- **Hyperparametry:** `LinearRegression()` v scikit-learn nemá žádné hyperparametry k ladění. Jedinou volbou je, zda zahrnout intercept  $w_0$  (parametr `fit_intercept`, standardně `True`).

#### 4.5 Problém kolinearita

- Kolinearita (angl. collinearity) nastává, když jsou příznaky (sloupce matice  $X$ ) silně korelované nebo téměř lineárně závislé.

| Problém               | Přímá inverze | SVD           | Ridge/Lasso |
|-----------------------|---------------|---------------|-------------|
| Numerická nestabilita | Velký problém | Vyřešeno      | Vyřešeno    |
| Vysoký rozptyl modelu | Problém       | Stále problém | Vyřešeno    |

Tabulka 3: Srovnání metod řešení lineární regrese při kolinearitě.

Tabulka 3 ukazuje srovnání a problémy při kolinearitě. I když SVD rozklad zlepší numerickou stabilitu, stále zůstává problém s vysokým rozptylem — malá změna v trénovacích datech způsobí velkou změnu v predikcích modelu. V praxi se to řeší pomocí Ridge, Lasso či Elastic Net regrese.

- Možné řešení kolinearita u lineární regrese:

1. Přigenerovat další data či odebrat existující (nemusí pomoci).
2. Snížit počet příznaků (feature selection, PCA) — chceme snížením počtu příznaků odebrat příznaky, které jsou (téměř) lineárně závislé.
3. Použití Ridge, Lasso či Elastic Net regrese, které mají regularizační členy, jež problém kolinearita většinou odstraní nebo dostatečně zmírní.

#### 4.6 Příklad výpočtu metody nejmenších čtverců

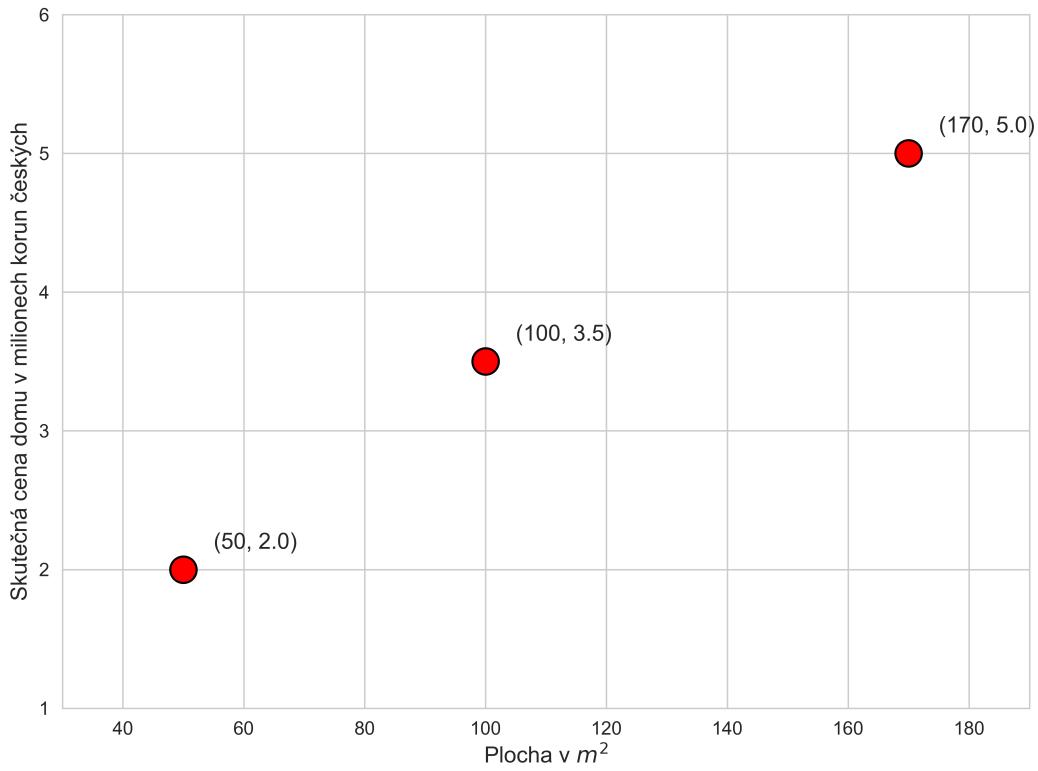
- Uvažme, že máme data se 3 domy a chceme predikovat jejich cenu podle plochy ( $m^2$ ). Potom vysvětlovaná proměnná  $\mathbf{Y}$  je:

$$\mathbf{Y} = (2.0, 3.5, 5.0)^T$$

a vektor ploch je:

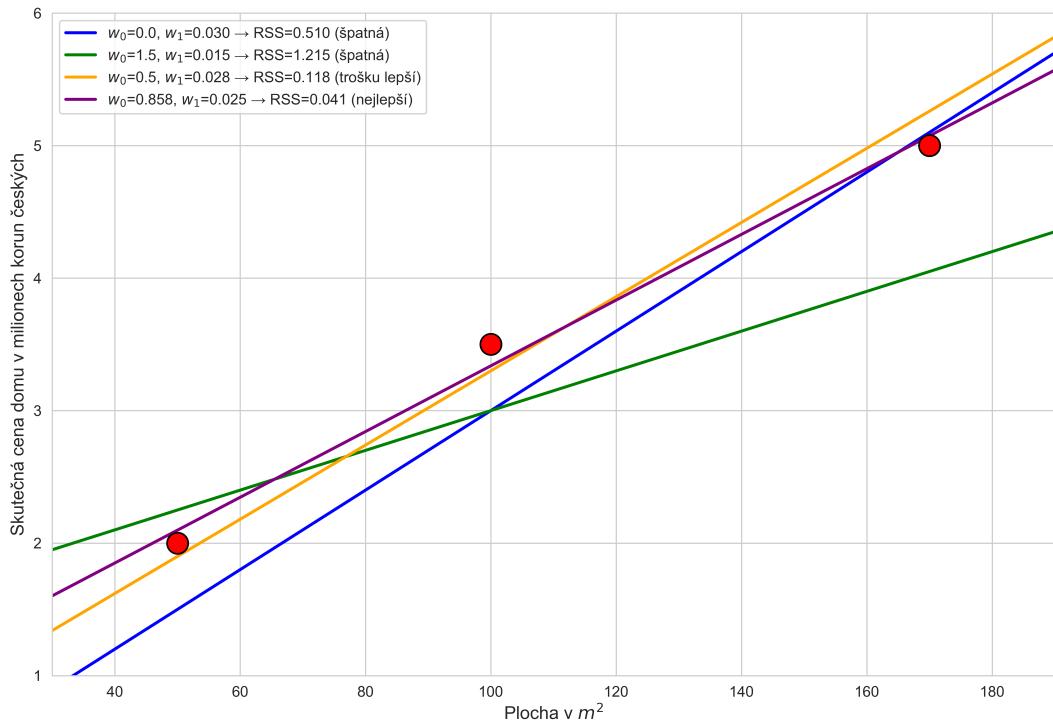
$$\mathbf{x} = (50, 100, 170)^T$$

- 3 domy reprezentované v grafu jsou vizualizované na obrázku 3.

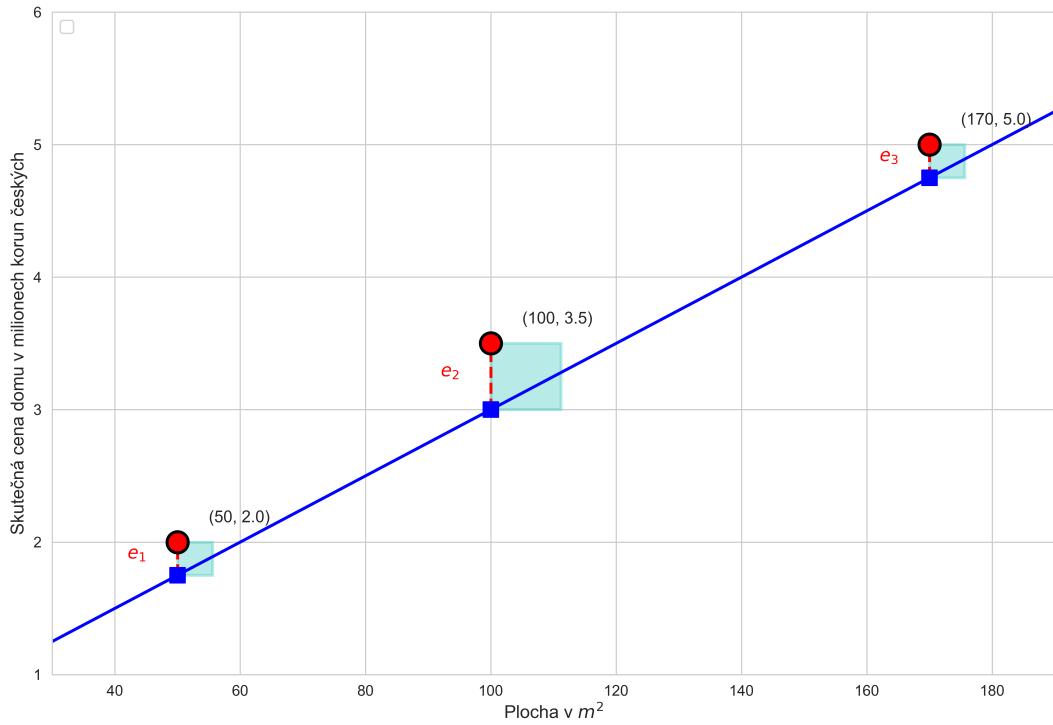


Obrázek 3: Každý bod  $(x_i, Y_i)$  představuje jeden dům v datasetu.

- Chceme nyní najít přímku  $\hat{Y} = w_0 + w_1 x$ , která bude co nejblíže ke všem bodům. Možné příklady přímek včetně té nejlepší možné reprezentuje [obrázek 4](#). Naopak [obrázek 5](#) reprezentuje residua, které se snažíme minimalizovat. Součet kvadrátů reziduí nazýváme residuální součet čtverců a značíme  $RSS(\mathbf{w})$ . Formálně tedy v našem ukázkovém příkladu platí:  $RSS(\mathbf{w}) = e_1^2 + e_2^2 + e_3^2$ .

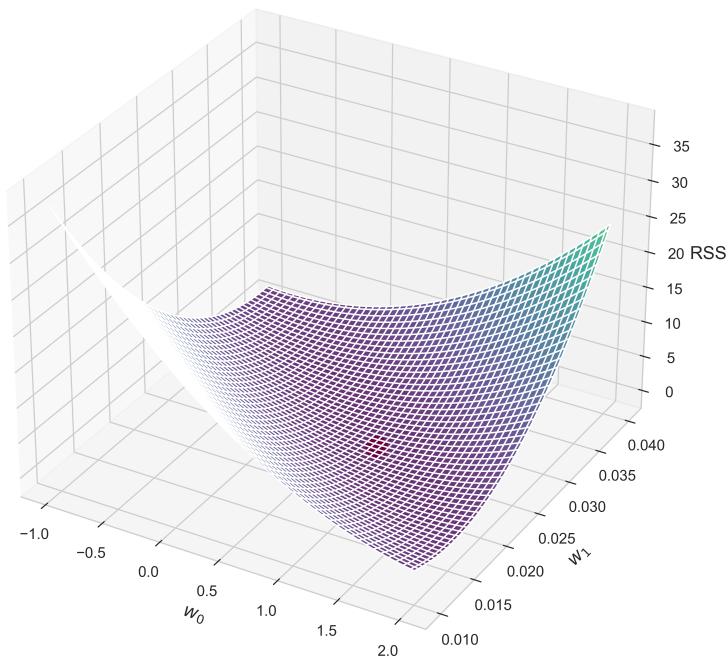


Obrázek 4: Porovnání různých přímek proložených daty.

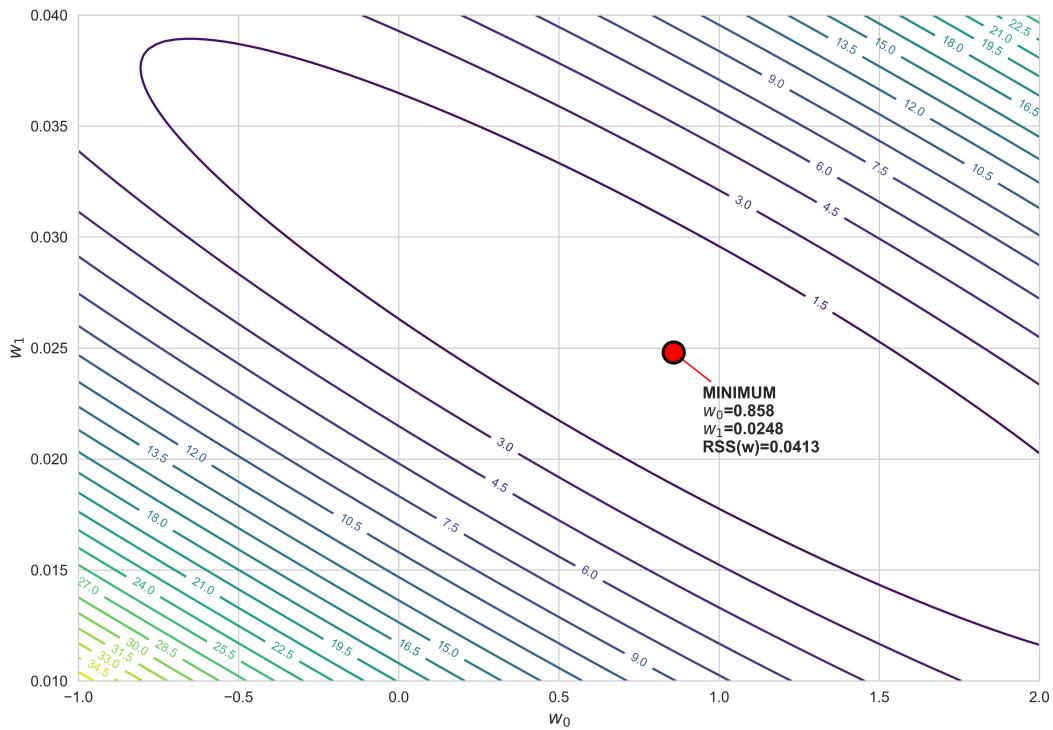


Obrázek 5: Vizualizace reziduí  $e_i = Y_i - \hat{Y}_i$ . Čtverce znázorňují hodnoty  $e_i^2$ , jejichž suma tvoří  $RSS(\mathbf{w})$ .

- Matematicky se snažíme najít minimum funkce  $RSS(\mathbf{w})$ . Funkci  $RSS(\mathbf{w})$  jako 3D plochu vizualizuje obrázek 6 a její vrstevnice obrázek 7.



Obrázek 6: Funkce  $RSS(w_0, w_1)$  jako 3D plocha. Minimum této funkce odpovídá optimálním parametrům  $\hat{w}_0, \hat{w}_1$ .



Obrázek 7: Vrstevnice (contour plot) funkce  $RSS(\mathbf{w})$ . Pohled shora na 3D plochu. Červený bod označuje minimum.

#### 4.6.1 Krok 1: Sestavení matice $\mathbf{X}$ a vektoru $\mathbf{Y}$

- Hledáme model ve tvaru  $\hat{Y} = w_0 + w_1x$ , proto matici  $\mathbf{X}$  sestavíme tak, aby první sloupec obsahoval samé jedničky (pro intercept  $w_0$ ) a druhý sloupec obsahoval hodnoty příznaku (plochy):

$$\mathbf{X} = \begin{pmatrix} 1 & 50 \\ 1 & 100 \\ 1 & 170 \end{pmatrix}, \quad \mathbf{Y} = \begin{pmatrix} 2.0 \\ 3.5 \\ 5.0 \end{pmatrix}, \quad \mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \end{pmatrix}$$

#### 4.6.2 Krok 2: Výpočet $\mathbf{X}^T\mathbf{X}$

- Transponovaná matice  $\mathbf{X}^T$  má tvar:

$$\mathbf{X}^T = \begin{pmatrix} 1 & 1 & 1 \\ 50 & 100 & 170 \end{pmatrix}$$

- Součin  $\mathbf{X}^T\mathbf{X}$ :

$$\begin{aligned} \mathbf{X}^T\mathbf{X} &= \begin{pmatrix} 1 & 1 & 1 \\ 50 & 100 & 170 \end{pmatrix} \begin{pmatrix} 1 & 50 \\ 1 & 100 \\ 1 & 170 \end{pmatrix} \\ &= \begin{pmatrix} 3 & 320 \\ 320 & 2500 + 10000 + 28900 \end{pmatrix} = \begin{pmatrix} 3 & 320 \\ 320 & 41400 \end{pmatrix} \end{aligned}$$

#### 4.6.3 Krok 3: Výpočet $\mathbf{X}^T\mathbf{Y}$

- Součin  $\mathbf{X}^T\mathbf{Y}$ :

$$\begin{aligned} \mathbf{X}^T\mathbf{Y} &= \begin{pmatrix} 1 & 1 & 1 \\ 50 & 100 & 170 \end{pmatrix} \begin{pmatrix} 2.0 \\ 3.5 \\ 5.0 \end{pmatrix} \\ &= \begin{pmatrix} 1 \cdot 2.0 + 1 \cdot 3.5 + 1 \cdot 5.0 \\ 50 \cdot 2.0 + 100 \cdot 3.5 + 170 \cdot 5.0 \end{pmatrix} \\ &= \begin{pmatrix} 10.5 \\ 100 + 350 + 850 \end{pmatrix} = \begin{pmatrix} 10.5 \\ 1300 \end{pmatrix} \end{aligned}$$

#### 4.6.4 Krok 4: Výpočet inverzní matice $(\mathbf{X}^T\mathbf{X})^{-1}$

Pro výpočet inverze matice  $\mathbf{X}^T\mathbf{X}$  použijeme Gauss-Jordanovu eliminaci na rozšířené matici  $(\mathbf{X}^T\mathbf{X} \mid \mathbf{E})$ :

$$\begin{aligned} \left( \begin{array}{cc|cc} 3 & 320 & 1 & 0 \\ 320 & 41400 & 0 & 1 \end{array} \right) &\sim \left( \begin{array}{cc|cc} 1 & \frac{320}{3} & \frac{1}{3} & 0 \\ 320 & 41400 & 0 & 1 \end{array} \right) \sim \left( \begin{array}{cc|cc} 1 & \frac{320}{3} & \frac{1}{3} & 0 \\ 0 & \frac{21800}{3} & -\frac{320}{3} & 1 \end{array} \right) \\ &\sim \left( \begin{array}{cc|cc} 1 & \frac{320}{3} & \frac{1}{3} & 0 \\ 0 & 1 & -\frac{320}{21800} & \frac{3}{21800} \end{array} \right) \sim \left( \begin{array}{cc|cc} 1 & 0 & \frac{41400}{21800} & -\frac{320}{21800} \\ 0 & 1 & -\frac{320}{21800} & \frac{3}{21800} \end{array} \right) \end{aligned}$$

Výsledná inverzní matice:

$$(\mathbf{X}^T\mathbf{X})^{-1} = \frac{1}{21800} \begin{pmatrix} 41400 & -320 \\ -320 & 3 \end{pmatrix}$$

#### 4.6.5 Krok 5: Řešení normální rovnice $\hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}$

Dosadíme do vzorce pro odhad vektoru vah:

$$\begin{aligned}\hat{\mathbf{w}} &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y} = \frac{1}{21800} \begin{pmatrix} 41400 & -320 \\ -320 & 3 \end{pmatrix} \begin{pmatrix} 10.5 \\ 1300 \end{pmatrix} \\ &= \frac{1}{21800} \begin{pmatrix} 434700 - 416000 \\ -3360 + 3900 \end{pmatrix} = \frac{1}{21800} \begin{pmatrix} 18700 \\ 540 \end{pmatrix} \\ &\approx \begin{pmatrix} 0.858 \\ 0.0248 \end{pmatrix}\end{aligned}$$

Výsledný model lineární regrese:

$$\hat{Y} = 0.858 + 0.0248 \cdot x$$

#### 4.6.6 Krok 6: Výpočet predikcí a reziduí

Výpočet predikcí  $\hat{Y}_i$  a reziduí  $e_i = Y_i - \hat{Y}_i$ :

$$\begin{aligned}\hat{Y}_1 &= 0.858 + 0.0248(50) = 2.098 \quad \rightarrow \quad e_1 = 2.0 - 2.098 = -0.098 \\ \hat{Y}_2 &= 0.858 + 0.0248(100) = 3.338 \quad \rightarrow \quad e_2 = 3.5 - 3.338 = 0.162 \\ \hat{Y}_3 &= 0.858 + 0.0248(170) = 5.074 \quad \rightarrow \quad e_3 = 5.0 - 5.074 = -0.074\end{aligned}$$

Residuální součet čtverců ( $RSS$ ):

$$RSS(\hat{\mathbf{w}}) = \sum_{i=1}^3 e_i^2 = (-0.098)^2 + (0.162)^2 + (-0.074)^2 \approx 0.041$$

#### 4.6.7 Shrnutí výsledků

- Výsledný model:**  $\hat{Y} = 0.858 + 0.0248 \cdot x$ .
- Interpretace:** Každý další  $m^2$  plochy zvyšuje cenu domu v průměru o 24 800 Kč ( $\hat{w}_1 \approx 0.0248$  mil. Kč). Intercept  $\hat{w}_0$  odpovídá teoretické základní ceně.

| Dům | $x_i$ ( $m^2$ ) | $Y_i$ (mil. Kč) | $\hat{Y}_i$ (predikce) | $e_i$ (chyba) |
|-----|-----------------|-----------------|------------------------|---------------|
| 1   | 50              | 2.0             | 2.098                  | -0.098        |
| 2   | 100             | 3.5             | 3.338                  | +0.162        |
| 3   | 170             | 5.0             | 5.074                  | -0.074        |

Tabulka 4: Shrnutí výsledků OLS regrese.

## 4.7 Geometrická interpretace metody nejmenších čtverců

**Otzávka ke zkoušce 4 (Geometrická interpretace metody nejmenších čtverců):** Geometrická interpretace metody nejmenších čtverců, normální rovnice, řešení. Regularita versus lineární nezávislost sloupců matice X.

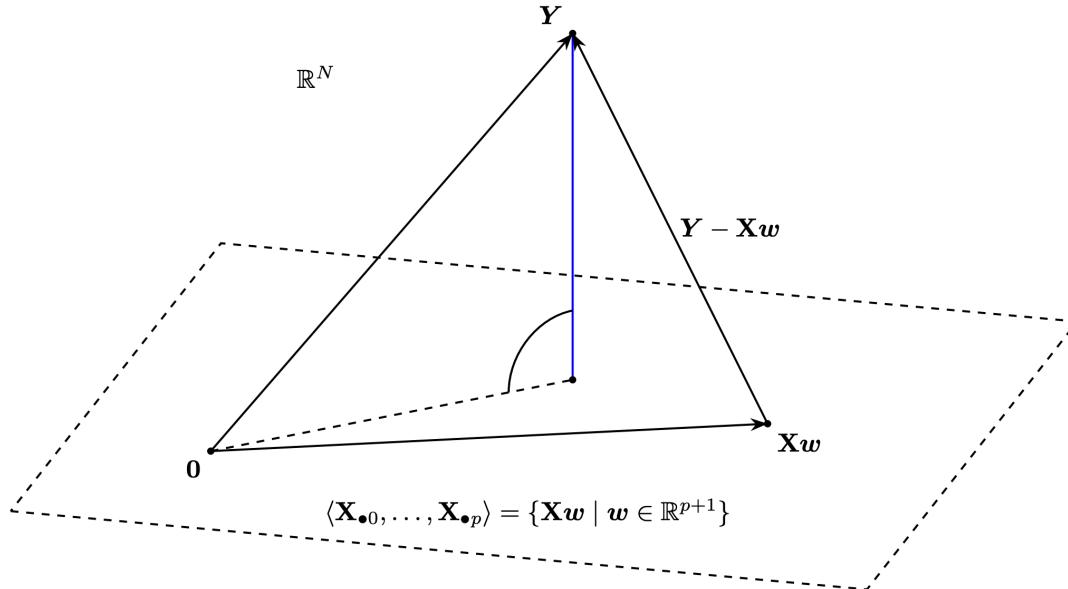
- Minimalizace  $RSS(\mathbf{w}) = \|\mathbf{Y} - \mathbf{X}\mathbf{w}\|^2$  je ekvivalentní minimalizaci  $\|\mathbf{Y} - \mathbf{X}\mathbf{w}\|$ .
- To znamená, že pro optimální  $\mathbf{w}$  je Euklidovská vzdálenost bodů  $\mathbf{Y}$  a  $\mathbf{X}\mathbf{w}$  v prostoru  $\mathbb{R}^N$  nejmenší možná.
- Označíme-li  $i$ -tý sloupec matice  $\mathbf{X}$  jako  $\mathbf{X}_{\bullet i}$ , můžeme si všimnout, že:

$$\mathbf{X}\mathbf{w} = w_0 \mathbf{X}_{\bullet 0} + w_1 \mathbf{X}_{\bullet 1} + \cdots + w_p \mathbf{X}_{\bullet p}$$

- Vektor  $\mathbf{X}\mathbf{w}$  je lineární kombinací sloupců matice  $\mathbf{X}$  s koeficienty  $w_0, \dots, w_p$ . Leží tedy v lineárním podprostoru prostoru  $\mathbb{R}^N$ , který je lineárním obalem  $p+1$  sloupců  $\mathbf{X}_{\bullet 0}, \dots, \mathbf{X}_{\bullet p}$ . Pro různé hodnoty  $\mathbf{w}$  pak vektor  $\mathbf{X}\mathbf{w}$  celý tento podprostor pokrývá, tj.:

$$\langle \mathbf{X}_{\bullet 0}, \dots, \mathbf{X}_{\bullet p} \rangle = \{ \mathbf{X}\mathbf{w} \mid \mathbf{w} \in \mathbb{R}^{p+1} \}$$

- Chceme-li minimalizovat vzdálenost  $\mathbf{Y}$  a  $\mathbf{X}\mathbf{w}$ , hledáme bod  $\mathbf{X}\mathbf{w}$  v podprostoru sloupců matice  $\mathbf{X}$ , který je k  $\mathbf{Y}$  nejblíže. Bod  $\mathbf{X}\mathbf{w}$  je k bodu  $\mathbf{Y}$  nejblíže, jestliže je vektor  $\mathbf{Y} - \mathbf{X}\mathbf{w}$  na ten podprostor kolmý, což reprezentuje modrý vektor na obrázku 8.



Obrázek 8: Geometrická interpretace metody nejmenších čtverců

- To znamená, že vektor rozdílu musí být kolmý na všechny vektory  $\mathbf{X}_{\bullet 0}, \dots, \mathbf{X}_{\bullet p}$ , které podprostor generují:

$$(\mathbf{X}_{\bullet i})^T (\mathbf{Y} - \mathbf{X}\mathbf{w}) = 0 \quad \text{pro } \forall i = 0, \dots, p$$

- To lze maticově zapsat jako:

$$\mathbf{X}^T (\mathbf{Y} - \mathbf{X}\mathbf{w}) = \mathbf{0} \iff \mathbf{X}^T \mathbf{Y} - \mathbf{X}^T \mathbf{X}\mathbf{w} = \mathbf{0}$$

Získali jsme tím starou známou **normální rovnici** a tedy stejné řešení. Z výše uvedených geometrických úvah navíc plyne, že pro jakékoli řešení  $\mathbf{w}$  normální rovnice je  $\|\mathbf{Y} - \mathbf{X}\mathbf{w}\|$ , a tedy i  $RSS(\mathbf{w})$ , nejmenší možné. Jakékoli řešení normální rovnice tedy dává globální minimum.

#### 4.8 Zajímavá videa na YouTube

- [Linear Regression, Clearly Explained!!! \(StatQuest with Josh Starmer\)](#) [27 min]
- [Orthogonal Projection Formulas \(Least Squares\) - Projection, Part 2 \(Sam Levey\)](#) [26 min]

## 5 Hřebenová regrese (Ridge Regression)

**Otzávka ke zkoušce 5 (Hřebenová regrese):** Regularizovaný reziduální součet čtverců, řešení. Modely bázových funkcí.

## 5.1 Hřebenová regrese: konstrukce

- Hřebenová regrese (angl. ridge regression) [Hoerl, Kennard (1970)] nebo taky  $L_2$  regularizace se k problému kolinearity staví zavedením penalizačního člena úměrného kvadrátu normy vektoru koeficientů  $\mathbf{w}$  s vynescháním interceptu.
- Minimalizuje se tedy **regularizovaný reziduální součet čtverců**

$$RSS_\lambda(\mathbf{w}) = \|\mathbf{Y} - \mathbf{X}\mathbf{w}\|^2 + \lambda \sum_{i=1}^p w_i^2$$

který závisí na parametru  $\lambda \geq 0$ .

- Pro  $\lambda = 0$  se dostává  $RSS_0(\mathbf{w}) = RSS(\mathbf{w})$  a dostáváme tedy obyčejnou metodu nejmenších čtverců.
- Pro  $\lambda > 0$  je vidět, že v minimu se bude cílit na takové vektory  $\mathbf{w}$ , které mají co nejmenší složky.
- Hodnota  $w_0$  interceptu se nijak nepenalizuje. Jedná se pouze o vertikální posun, který zajišťuje předpoklad  $\mathbb{E}\varepsilon = 0$  modelu a je tedy vhodné ho neomezovat.
- Stále platí, že model pro  $Y$  v bodě  $\mathbf{x}$  je  $Y = w_0 + w_1x_1 + \dots + w_px_p + \varepsilon$ .
- Zavedeme-li matici

$$\mathbf{I}' = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \in \mathbb{R}^{p+1,p+1},$$

lze psát

$$RSS_\lambda(\mathbf{w}) = \sum_{i=1}^N (Y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \lambda \sum_{i=1}^p w_i^2 = \|\mathbf{Y} - \mathbf{X}\mathbf{w}\|^2 + \lambda \sum_{i=1}^p w_i^2 = \|\mathbf{Y} - \mathbf{X}\mathbf{w}\|^2 + \lambda \mathbf{w}^T \mathbf{I}' \mathbf{w}.$$

- Hledá se minimum, proto sestrojí se **gradient**  $RSS_\lambda(\mathbf{w})$ :

$$\begin{aligned} \nabla RSS_\lambda(\mathbf{w}) &= \frac{\partial RSS_\lambda(\mathbf{w})}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}} (\|\mathbf{Y} - \mathbf{X}\mathbf{w}\|^2 + \lambda \mathbf{w}^T \mathbf{I}' \mathbf{w}) \\ &= \frac{\partial}{\partial \mathbf{w}} (\mathbf{Y}^T \mathbf{Y} - 2\mathbf{Y}^T \mathbf{X}\mathbf{w} + \mathbf{w}^T \mathbf{X}^T \mathbf{X}\mathbf{w} + \lambda \mathbf{w}^T \mathbf{I}' \mathbf{w}) \\ &= -2\mathbf{X}^T \mathbf{Y} + 2\mathbf{X}^T \mathbf{X}\mathbf{w} + 2\lambda \mathbf{I}' \mathbf{w} \\ &= -2\mathbf{X}^T (\mathbf{Y} - \mathbf{X}\mathbf{w}) + 2\lambda \mathbf{I}' \mathbf{w} \end{aligned}$$

- Ekvivalent normální rovnice je tedy

$$\mathbf{X}^T \mathbf{Y} - \mathbf{X}^T \mathbf{X}\mathbf{w} - \lambda \mathbf{I}' \mathbf{w} = \mathbf{0}.$$

- **Hessova matice** je dále

$$\begin{aligned} \nabla^2 RSS_\lambda(\mathbf{w}) &= \frac{\partial}{\partial \mathbf{w}} (\nabla RSS_\lambda(\mathbf{w})) = \frac{\partial}{\partial \mathbf{w}} (-2\mathbf{X}^T \mathbf{Y} + 2\mathbf{X}^T \mathbf{X}\mathbf{w} + 2\lambda \mathbf{I}' \mathbf{w}) \\ &= 2\mathbf{X}^T \mathbf{X} + 2\lambda \mathbf{I}' \\ &= 2(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}'). \end{aligned}$$

- $\forall \mathbf{v} \in \mathbb{R}^{p+1}$ ,  $\mathbf{v} \neq \mathbf{0}$  a  $\lambda > 0$  platí

$$\mathbf{v}^T (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}') \mathbf{v} = (\mathbf{X}\mathbf{v})^T (\mathbf{X}\mathbf{v}) + \lambda \mathbf{v}^T \mathbf{I}' \mathbf{v} = \|\mathbf{X}\mathbf{v}\|^2 + \lambda \sum_{i=1}^p v_i^2 > 0,$$

protože pro  $\mathbf{v} = (v_0, 0, \dots, 0)^T \neq \mathbf{0}$  platí  $\mathbf{X}\mathbf{v} = (v_0, \dots, v_0)^T \neq \mathbf{0}$ .

- Hessova matice je tedy pozitivně definitní a matice  $\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}'$  je vždy regulární pro  $\lambda > 0$ .
- Pro  $\lambda > 0$  tak vždy existuje jednoznačné řešení normální rovnice

$$\hat{\mathbf{w}}_\lambda = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}')^{-1} \mathbf{X}^T \mathbf{Y}$$

a odpovídá globálnímu minimu  $RSS_\lambda$ .

- Predikce v bodě  $\mathbf{x}$  je potom opět  $\hat{Y} = \mathbf{x}^T \hat{\mathbf{w}}_\lambda$ .

## 5.2 Modely bázových funkcí

- Doposud se uvažoval pouze jednoduchý lineární model  $Y = \mathbf{x}^T \mathbf{w} + \varepsilon$ .
- Jak rozšířit možnosti za obzor linearity?
- Základní rozšíření spočívá v nahrazení původních příznaků jejich transformovanými variantami.
- Pro  $M \in \mathbb{N}$  vezměme  $M$  funkcí  $\varphi_1, \dots, \varphi_M$  z  $\mathbb{R}^p$  do  $\mathbb{R}$ , které reprezentují transformace příznaků  $X$ , a nazveme je **bázové funkce** (angl. basis functions).
- K těmto funkcím se přidá  $\varphi_0(\mathbf{x}) = 1$  (intercept) a poskládají se do vektorové funkce  $\varphi : \mathbb{R}^p \rightarrow \mathbb{R}^{M+1}$  vztahem  $\varphi(\mathbf{x}) = (1, \varphi_1(\mathbf{x}), \dots, \varphi_M(\mathbf{x}))^T$ .
- Jako model vztahu  $Y$  a  $\mathbf{x}$  budeme uvažovat lineární model:

$$Y = \sum_{j=0}^M w_j \varphi_j(\mathbf{x}) + \varepsilon = \varphi(\mathbf{x})^T \mathbf{w} + \varepsilon.$$

- Další postup je analogický jako předtím.
- Predikce:  $\hat{Y} = \varphi(\mathbf{x})^T \hat{\mathbf{w}}_\lambda$ .

### Příklady bázových funkcí:

- $\varphi(\mathbf{x}) = x_i$  — přímo jednotlivé příznaky.
- $\varphi(\mathbf{x}) = x_i^2, \varphi(\mathbf{x}) = x_k x_\ell$  — mocniny příznaků a jejich různé součiny, polynomiální regrese.
- $\varphi(\mathbf{x}) = \log(x_i), \sqrt{x_i}, \sin(x_i)$  — nelineární transformace jednotlivých příznaků.
- $\varphi(\mathbf{x}) = \mathbf{1}_{(a,b)}(x_i)$ , kde  $\mathbf{1}_A(\mathbf{x}) = 1$  pokud  $\mathbf{x} \in A$  a 0 jinak — indikátory množin.
- $\varphi(\mathbf{x}) = h(\|\mathbf{x} - \mathbf{x}_i\|)$ , kde  $\mathbf{x}_i$  je  $i$ -tý trénovací bod a  $h$  je nějaká funkce — tzv. radiální bázové funkce centrované v bodech trénovací množiny.

## 6 Statistické vlastnosti modelů

**Otázka ke zkoušce 6 (Statistické vlastnosti modelů):** Rozklad očekávané chyby modelu, bias-variance tradeoff. Nestrannost odhadu v lineární regresi metodou nejmenších čtverců.

### 6.1 Rozklad očekávané chyby modelu

- Budeme předpokládat nezávislost trénovacích a testovacích dat, tj. nezávislost  $\mathbf{Y}$  a  $Y$ , a v důsledku tedy nezávislost  $\hat{Y}$  a  $Y$ .
- Pro očekávanou chybu tedy platí

$$\begin{aligned} \mathbb{E}L(Y, \hat{Y}) &= \mathbb{E}(Y - \hat{Y})^2 = \mathbb{E}(Y - \mathbb{E}Y + \mathbb{E}Y - \hat{Y})^2 \\ &= \mathbb{E}(Y - \mathbb{E}Y)^2 + 2\mathbb{E}[(Y - \mathbb{E}Y)(\mathbb{E}Y - \hat{Y})] + \mathbb{E}(\hat{Y} - \mathbb{E}Y)^2 \\ &= \mathbb{E}(Y - \mathbb{E}Y)^2 + \mathbb{E}(\hat{Y} - \mathbb{E}Y)^2. \end{aligned}$$

Označíme-li  $\text{var } Y = \text{var } \varepsilon = \sigma^2$ , dostáváme

$$\mathbb{E}L(Y, \hat{Y}) = \sigma^2 + \mathbb{E}(\hat{Y} - \mathbb{E}Y)^2.$$

- První člen odpovídá **neodstranitelné chybě**, která je dána náhodností v modelu. Tato chyba se nazývá Bayesovská (angl. **Bayes error**). Druhý člen se značí  $MSE(\hat{Y})$  a nazývá **střední kvadratická chyba odhadu** parametru  $\mathbb{E}Y$  (angl. **mean squared error**).
- Pro  $MSE(\hat{Y})$  dále platí

$$\begin{aligned}
 MSE(\hat{Y}) &= \mathbb{E}(\hat{Y} - \mathbb{E}Y)^2 = \mathbb{E}(\mathbb{E}\hat{Y} - \mathbb{E}Y + \hat{Y} - \mathbb{E}\hat{Y})^2 \\
 &= \mathbb{E}(\mathbb{E}\hat{Y} - \mathbb{E}Y)^2 + \mathbb{E}(\hat{Y} - \mathbb{E}\hat{Y})^2 + 2\mathbb{E}(\hat{Y} - \mathbb{E}\hat{Y})(\mathbb{E}\hat{Y} - \mathbb{E}Y) \\
 &= (\mathbb{E}\hat{Y} - \mathbb{E}Y)^2 + \mathbb{E}(\hat{Y} - \mathbb{E}\hat{Y})^2 + 2 \cdot 0 \cdot (\mathbb{E}\hat{Y} - \mathbb{E}Y) \\
 &= (\mathbb{E}\hat{Y} - \mathbb{E}Y)^2 + \text{var } \hat{Y} = (\text{bias } \hat{Y})^2 + \text{var } \hat{Y}
 \end{aligned}$$

kde  $\text{bias } \hat{Y} = \mathbb{E}Y - \mathbb{E}\hat{Y}$  značí **vychýlení odhadu** (angl. **bias**).

- Dohromady tedy máme finální dekompozici očekávané chyby jako

$$\mathbb{E}L(Y, \hat{Y}) = \sigma^2 + (\text{bias } \hat{Y})^2 + \text{var } \hat{Y}.$$

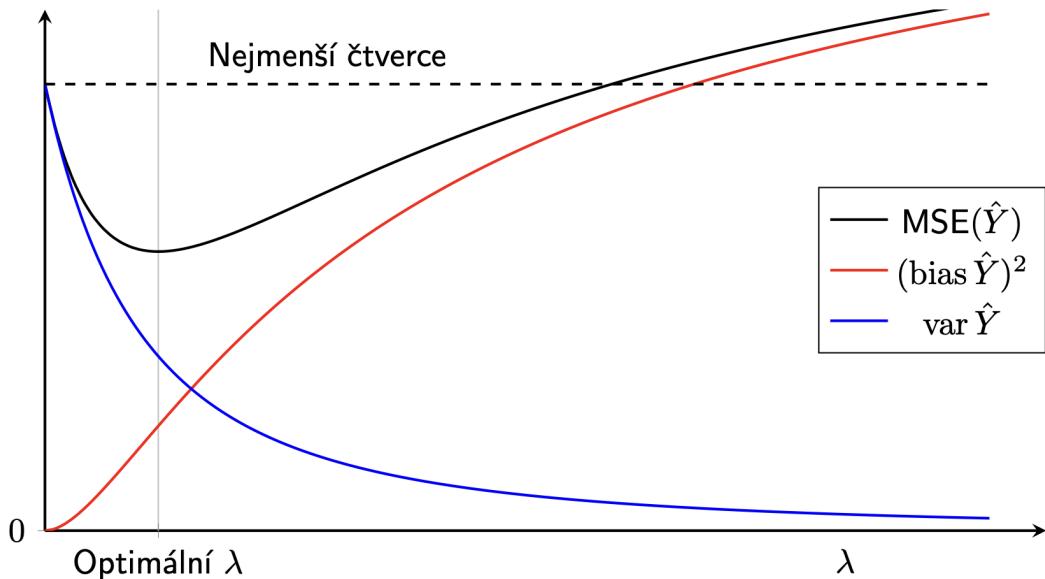
Očekávaná chyba modelu je tedy součtem **neodstranitelné chyby**, **kvadrátu vychýlení odhadu** a **rozptylu odhadu**.

## 6.2 Bias-variance tradeoff (ilustrace na hřebenové regresi)

U hřebenové regrese lze ukázat, že (hodně zjednodušeně) platí

$$(\text{bias } \hat{Y})^2 \sim \left(1 - \frac{1}{1+\lambda}\right)^2 \quad \text{a} \quad \text{var } \hat{Y} \sim \left(\frac{1}{1+\lambda}\right)^2.$$

To znamená, že s rostoucím  $\lambda$  vychýlení roste a rozptyl klesá. Takového chování v závislosti na hyperparametrech modelu je typické a nazývá se **bias-variance tradeoff**.



Obrázek 9: Závislost  $\text{MSE}(\hat{Y})$ ,  $(\text{bias } \hat{Y})^2$  a  $\text{var } \hat{Y}$  na parametru regularizace  $\lambda$ .

### 6.3 Nestrannost odhadu v lineární regresi metodou nejmenších čtverců

- Odhad vektoru parametrů  $\hat{\mathbf{w}}_\lambda = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}$  je náhodný vektor.
- Je to tedy **bodový odhad** vektoru parametrů  $\mathbf{w}$  a je příkladem tzv. statistiky.
- **Věta:** Odhad  $\hat{\mathbf{w}}_{OLS}$  získaný metodou nejmenších čtverců je za předpokladu  $\mathbb{E}\boldsymbol{\varepsilon} = \mathbf{0}$  nestranný, tj.  $\mathbb{E}\hat{\mathbf{w}}_{OLS} = \mathbf{w}$ .
- **Důkaz:** Z linearity střední hodnoty plyne:

$$\mathbb{E}\mathbf{Y} = \mathbb{E}(\mathbf{X}\mathbf{w} + \boldsymbol{\varepsilon}) = \mathbf{X}\mathbf{w} + \mathbb{E}\boldsymbol{\varepsilon} = \mathbf{X}\mathbf{w}.$$

Dále platí:

$$\begin{aligned}\mathbb{E}\hat{\mathbf{w}}_{OLS} &= \mathbb{E}[(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}] = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbb{E}\mathbf{Y} \\ &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{X}\mathbf{w} = \mathbf{w}.\end{aligned}$$

- Z nestrannosti odhadu vektoru parametrů  $\hat{\mathbf{w}}_{OLS}$  dále plyne nestrannost odhadu  $\hat{Y}$  v bodě  $\mathbf{x}$ .
- K tomu je třeba si uvědomit, že pomocí  $\hat{Y} = \mathbf{x}^T \hat{\mathbf{w}}_{OLS}$  se snažíme predikovat skutečnou hodnotu  $Y$ , ale ze statistického pohledu se jedná o **bodový odhad střední hodnoty**  $\mathbb{E}Y = \mathbf{x}^T \mathbf{w} + \mathbb{E}\boldsymbol{\varepsilon} = \mathbf{x}^T \mathbf{w}$ .
- Z předchozí věty plyne:  

$$\mathbb{E}\hat{Y} = \mathbb{E}\mathbf{x}^T \hat{\mathbf{w}}_{OLS} = \mathbf{x}^T \mathbb{E}\hat{\mathbf{w}}_{OLS} = \mathbf{x}^T \mathbf{w} = \mathbb{E}Y$$
  
 a  $\hat{Y}$  je tedy nestranným odhadem  $\mathbb{E}Y$ .
- To samozřejmě znamená, že  

$$\text{bias } \hat{Y} = \mathbb{E}\hat{Y} - \mathbb{E}Y = 0,$$
  
 tj. vychýlení je 0.

## 7 Logistická regrese

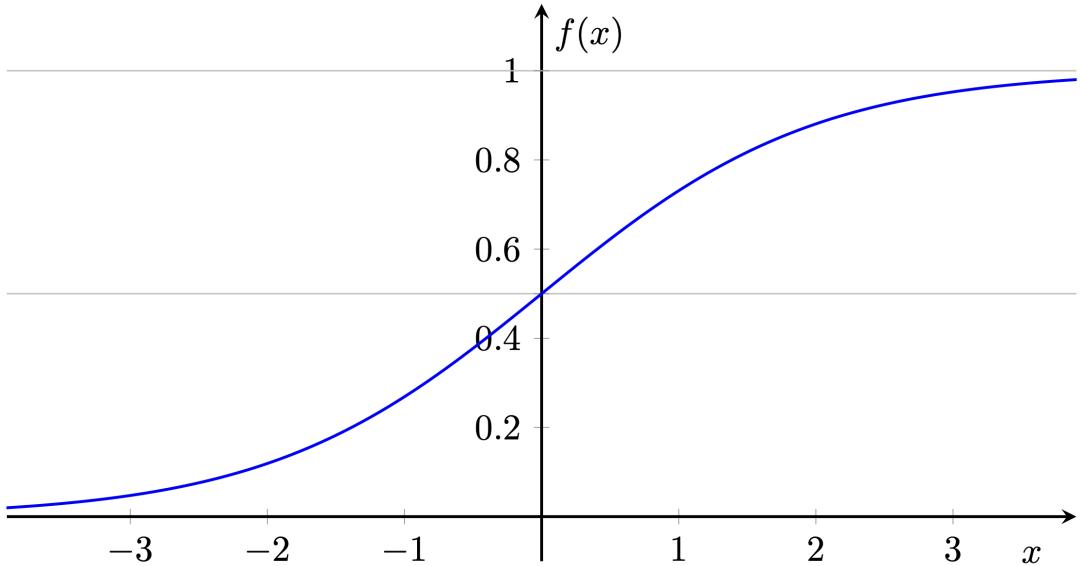
**Otzáka ke zkoušce 7 (Logistická regrese):** Použití pro binární klasifikaci, jak funguje model, vlastnosti.

### 7.1 Použití pro binární klasifikaci

- Metoda určená pro **klasifikaci** (vysvětlovaná proměnná  $Y \in \{0, 1\}$ ).
- Základní myšlenka: namísto hodnoty  $Y \in \{0, 1\}$  se snažíme predikovat pravděpodobnost  $P(Y = 1)$  z intervalu  $[0, 1]$ .
- Hledáme funkční předpis, který pro dané hodnoty  $x_i$  příznaků  $X_i$  a dané hodnoty koeficientů  $w_i$  vrátí číslo z intervalu  $[0, 1]$ , které bude odhadem pravděpodobnosti rýmy ( $Y = 1$ ).

### 7.2 Logistická funkce (Sigmoida)

- $f(x) = \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}$
- $D_f = \mathbb{R}, H_f = (0, 1)$
- Ostře rostoucí funkce



Obrázek 10: Graf sigmoidy

### 7.3 Způsob výpočtu

- $\mathbf{x} = (x_0, x_1, \dots, x_p)$  - vektor příznaků
- $\mathbf{w} = (w_0, w_1, \dots, w_p)$  - vektor koeficientů
- $\mathbf{w}^T \mathbf{x}$  spočítáme a dosadíme do sigmoidy:  $P(Y = 1 | \mathbf{x}, \mathbf{w}) = \frac{e^{\mathbf{w}^T \mathbf{x}}}{1 + e^{\mathbf{w}^T \mathbf{x}}}$
- Predikce:  $\hat{Y} = 1$  pokud  $P(Y = 1) > \frac{1}{2}$ , jinak  $\hat{Y} = 0$

### 7.4 Logistická regrese: vlastnosti

- Nevyžaduje lineární vztah  $Y$  na příznacích
- Náchylná k overfittingu (zejména pokud je příliš složitá a trénovací data jsou omezená)
- Hodnoty  $\mathbf{w}$  odhadujeme pomocí metody MLE (viz další kapitola)

### 7.5 Maximálně věrohodný odhad (Maximum Likelihood Estimation)

**Otzávka ke zkoušce 8 (Logistická regrese):** Logistická regrese jako MLE odhad: sestavení optimalizační úlohy pro trénování, myšlenka MLE odhadů.

- Základní myšlenka MLE je, že máme trénovací data a chceme najít parametry modelu. MLE říká: "Vyber takové parametry, aby byla trénovací data, která vidíš, co nejpravděpodobnější."
- Odhad metodou MLE pak odpovídá hodnotě  $\mathbf{w}$ , pro kterou jsou trénovací data nejpravděpodobnější možná.
- **Pravděpodobnosti pro logistickou regresi:**

Označme pro úsporu místa:

$$p_1(\mathbf{x}, \mathbf{w}) = P(Y = 1 | \mathbf{x}, \mathbf{w}) = \frac{e^{\mathbf{w}^T \mathbf{x}}}{1 + e^{\mathbf{w}^T \mathbf{x}}}$$

$$p_0(\mathbf{x}, \mathbf{w}) = P(Y = 0 | \mathbf{x}, \mathbf{w}) = 1 - \frac{e^{\mathbf{w}^T \mathbf{x}}}{1 + e^{\mathbf{w}^T \mathbf{x}}} = \frac{1}{1 + e^{\mathbf{w}^T \mathbf{x}}}$$

- **Pravděpodobnost datového bodu:**

Máme-li  $i$ -tý datový bod s hodnotou vysvětlované proměnné  $Y_i$  a s hodnotami příznaků  $\mathbf{x}_i = (x_{i;0}, x_{i;1}, \dots, x_{i;p})$ , lze pro zadané hodnoty parametrů  $\mathbf{w}$  označit pravděpodobnost tohoto datového bodu jako  $p_{Y_i}(\mathbf{x}_i, \mathbf{w})$ .

- **Notace pro trénovací data:**

Předpokládejme, že máme  $N$  bodů v trénovacích datech. Zapíšeme je do vektoru

$$\mathbf{Y} = (Y_1, Y_2, \dots, Y_N)^T \in \mathbb{R}^N$$

a do matice

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_N^T \end{pmatrix} = \begin{pmatrix} 1 & x_{1;1} & x_{1;2} & \cdots & x_{1;p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N;1} & x_{N;2} & \cdots & x_{N;p} \end{pmatrix}$$

kde  $x_{i;0} = 1$  je intercept.

- **Věrohodnostní funkce** (angl. likelihood function):

Vyjádříme pravděpodobnost konkrétních trénovacích dat při parametrech  $\mathbf{w}$ . Předpokládáme (většinou oprávněně), že jednotlivé datové body jsou navzájem nezávislé, pravděpodobnost lze tedy napsat jako součin pravděpodobností jednotlivých bodů:

$$L(\mathbf{w}) = \prod_{i=1}^N p_{Y_i}(\mathbf{x}_i, \mathbf{w})$$

což je reálná funkce  $p + 1$  proměnných  $\mathbb{R}^{p+1} \rightarrow \mathbb{R}$  vyjadřující pravděpodobnost trénovacích dat pro danou hodnotu parametrů  $\mathbf{w}$ .

- **Logaritmus věrohodnostní funkce:**

Místo součinu pravděpodobností derivujeme jejich logaritmus (logaritmus je ostře rostoucí funkce, takže extrémy jsou ve stejných bodech):

$$\begin{aligned} \ell(\mathbf{w}) &= \ln L(\mathbf{w}) = \sum_{i=1}^N \ln p_{Y_i}(\mathbf{x}_i, \mathbf{w}) \\ &= \sum_{i=1}^N (Y_i \ln p_1(\mathbf{x}_i, \mathbf{w}) + (1 - Y_i) \ln p_0(\mathbf{x}_i, \mathbf{w})) \\ &= \sum_{i=1}^N \left( Y_i \ln \left( \frac{e^{\mathbf{w}^T \mathbf{x}_i}}{1 + e^{\mathbf{w}^T \mathbf{x}_i}} \right) + (1 - Y_i) \ln \left( \frac{1}{1 + e^{\mathbf{w}^T \mathbf{x}_i}} \right) \right) \\ &= \sum_{i=1}^N \left( Y_i \mathbf{w}^T \mathbf{x}_i - \ln(1 + e^{\mathbf{w}^T \mathbf{x}_i}) \right) \end{aligned}$$

- **Gradient logaritmu věrohodnostní funkce:**

Stejně jako v případě lineární regrese se najde gradient, tj. vektor složený z parciálních derivací podle všech proměnných  $w_0, w_1, \dots, w_p$ :

$$\frac{\partial \ell}{\partial w_j}(\mathbf{w}) = \sum_{i=1}^N x_{i;j} (Y_i - p_1(\mathbf{x}_i, \mathbf{w})), \quad j = 0, 1, \dots, p$$

Pomocí maticového násobení lze pak gradient napsat ve tvaru:

$$\nabla \ell(\mathbf{w}) = \mathbf{X}^T (\mathbf{Y} - \mathbf{P})$$

kde  $\mathbf{P} = (p_1(\mathbf{x}_1, \mathbf{w}), p_1(\mathbf{x}_2, \mathbf{w}), \dots, p_1(\mathbf{x}_N, \mathbf{w}))^T$ .

- **Hledání maxima:**

Teorie říká, že maximum bychom měli nalézt mezi řešeními rovnice "gradient se rovná nule", tedy

$$\nabla \ell(\mathbf{w}) = \mathbf{X}^T (\mathbf{Y} - \mathbf{P}) = \mathbf{0}$$

- **Problém:**

Na rozdíl od lineární regrese neumíme najít explicitní řešení, tedy neexistuje vzorec, do kterého bychom dosadili trénovací data a vypadly by nám z něho hodnoty koeficientů  $w$ .

- **Numerické metody:**

Je nutné použít numerické approximativní metody. Používají se bud' vícerozměrná verze Newtonovy metody, nebo gradientní vzestup: v obou případech se konstruuje posloupnost  $w^{(0)}, w^{(1)}, w^{(2)}, \dots$  o které lze předpokládat, že konverguje k nějakému lokálnímu maximu. Pro logistickou regresi lze ukázat, že pokud existuje lokální maximum, je jediné a je to hledané globální maximum.

## 7.6 Závěrečné poznámky

- Celá metoda stojí na předpokladu, že se chování dat dá zachytit ve tvaru daném sigmoidou.

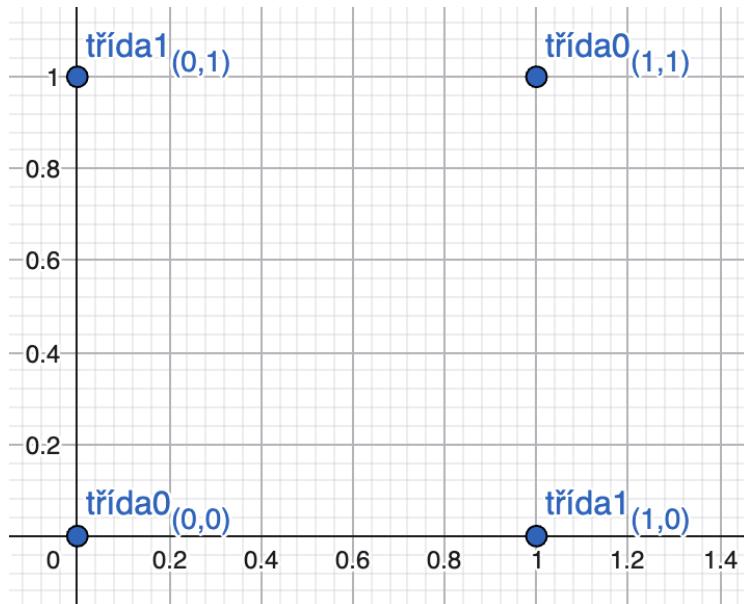
**Příklad 1** kruhová hranice:

- Třída 1: body **uvnitř** kruhu.
- Třída 0: body **mimo** kruh.

Logistická regrese selže! hranice je nelineární.

**Příklad 2** XOR problém:

- $(0,0) \rightarrow$  třída 0
- $(0,1) \rightarrow$  třída 1
- $(1,0) \rightarrow$  třída 1
- $(1,1) \rightarrow$  třída 0



Obrázek 11: Vizualizace bodů (Problém 2)

Logistická regrese opět selže! Protože nelze najít lineární hranici, která by třídy oddělila. Na toto potřebujeme nelineární model (např. neural network)

Problém 1 (XOR) nebo Problém 2 (kruh) **lze vyřešit i logistickou regresí, pokud provedeme transformaci příznaků.** Např. přidáním příznaku  $x_1^2 + x_2^2$  (pro kruh) nebo  $x_1 \cdot x_2$  (pro XOR)

## 8 Ensemble metody (Ensemble Methods)

**Otázka ke zkoušce 9 (Ensemble metody: bagging, boosting a náhodné lesy):** Rozdíl mezi baggingem a boostingem. Náhodné lesy a jejich hyperparametry.

### 8.1 Ensemble metody: základní myšlenka

- Základní myšlenka spočívá v tom, že namísto jednoho modelu (např. rozhodovacího stromu) použijeme **více modelů** a jejich predikce nějakým způsobem zkombinujeme do finálního rozhodnutí.
- My si ukážeme dva nejobvyklejší způsoby: **bagging** (bootstrap aggregating) a **boosting**.
- Každý z těchto dvou přístupů si ilustrujeme na nejznámějších reprezentantech, v obou případech spočívajících ve skládání rozhodovacích stromů, které už známe.
- Tito dva reprezentanti jsou: **náhodný les** (angl. Random Forest) pro bagging a **AdaBoost** (Adaptive Boosting) pro boosting.

### 8.2 Bagging vs. Boosting

- **Při baggingu** vybíráme do bagů (množin  $\mathcal{D}_i$ ) data náhodně.
- **Při boostingu** tuto metodu upravíme tak, že data, co byla dříve špatně klasifikována mají větší pravděpodobnost být vybrána.
- **Shrnutí rozdílů:**

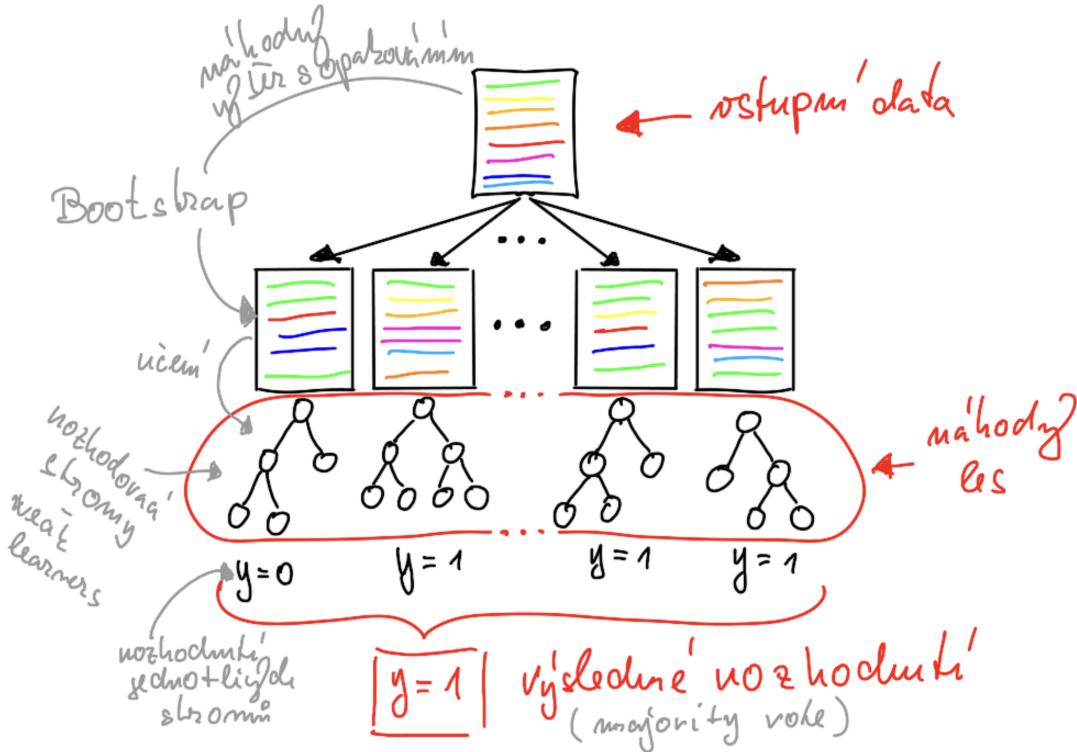
| Vlastnost         | Bagging                 | Boosting                                      |
|-------------------|-------------------------|---|
| Konstrukce modelů | Paralelní (nezávislé)   | Sekvenční (závislé)                           |
| Datové sady       | Bootstrapem, rovnoměrně | S váhami, preferují špatně klasifikovaná data |
| Redukuje          | Variance (rozptyl)      | Bias (vychýlení)                              |
| Podmodely         | Plně vyvinuté (deep)    | Slabé modely (weak learners, shallow)         |

Tabulka 5: Srovnání metod bagging a boosting.

### 8.3 Bagging: náhodné lesy

- **Náhodný les** (angl. Random Forest) pro klasifikaci je ensemble metoda založená na **baggingu** (bootstrap aggregating).
- Náhodný les pro klasifikaci je základní myšlenka.
- Pro jednoduchost předpokládejme binární klasifikační problém ( $Y$  je 0 nebo 1).
- Ze vstupního trénovacího datasetu  $\mathcal{D}$  vytvoříme  $n$  datasetů  $\mathcal{D}_1, \dots, \mathcal{D}_n$  obvykle stejně velkých pomocí metody **bootstrap** neboli pomocí výběru s opakováním.
- Na každém datasetu  $\mathcal{D}_i$  naučíme rozhodovací strom tak, jak jsem si předvedli v minulé přednášce. Stromy se obvykle nechávají narůst hluboké (často bez omezení hloubky), aby měly nízké vychýlení (bias). Bagging pak redukuje jejich vysokou varianci průměrováním. Stromy označme  $T_1, \dots, T_n$ .
- Každý datový bod (řádek z tabulky s daty z  $\mathcal{D}$ ) proženeme všemi stromy  $T_1, \dots, T_n$  a od každého z nich si uložíme rozhodnutí  $Y_1, \dots, Y_n$ .
- Všechny tyto stromy tvoří **náhodný les** a jeho finální rozhodnutí o hodnotě  $Y$  je dané **většinovým rozhodnutím stromů** — při klasifikaci nejčastější hodnota.

## 8.4 Náhodný les: vizualizace



Obrázek 12: Náhodný les pro klasifikaci — základní myšlenka

Na obrázku 12 vidíme vstupní data (nahoře), z nichž bootstrapem vytvoříme několik datasetů (střední řada). Na každém z nich natrénujeme rozhodovací strom. Finální predikce náhodného lesa je dána většinovým hlasováním (majority vote) všech stromů.

## 8.5 Bootstrap

- Ukážeme si, jak funguje bootstrap na jednoduchém příkladu:

| id | rýmička | pohlaví | $> 39^{\circ}\text{C}$ | vstal(a)? |
|----|---------|---------|------------------------|-----------|
| 1  | ano     | muž     | ne                     | ne        |
| 2  | ne      | žena    | ano                    | ano       |
| 3  | ne      | muž     | ne                     | ano       |
| 4  | ano     | žena    | ano                    | ne        |

Tabulka 6: Původní trénovací data pro ilustraci metody bootstrap.

- Chceme-li vytvořit "bootstrapem" dataset velikosti pět, pětkrát si náhodně vybereme řádek z tabulky s tím, že řádky v našem výběru mohou opakovat.
- Vybereme-li např. řádky s id 1, 4, 3, 3, 1, dostaneme dataset:

## 8.6 Predikce náhodného lesa

- **V případě regresního problému** (spojité  $Y$ ) se postupuje analogicky jako u regresního rozhodovacího stromu: predikce náhodného lesa se bere jako **průměr z predikcí jednotlivých stromů** lesa:

$$\hat{Y} = \frac{1}{n} \sum_{i=1}^n \hat{Y}_i$$

| <b>id</b> | <b>rýmička</b> | <b>pohlaví</b> | <b>&gt; 39°C</b> | <b>vstal(a)?</b> |
|-----------|----------------|----------------|------------------|------------------|
| 1         | ano            | muž            | ne               | ne               |
| 4         | ano            | žena           | ano              | ne               |
| 3         | ne             | muž            | ne               | ano              |
| 3         | ne             | muž            | ne               | ano              |
| 1         | ano            | muž            | ne               | ne               |

Tabulka 7: Dataset vytvořený metodou bootstrap (výběr s opakováním).

- **Pro klasifikaci:** Implementace `RandomForestClassifier` ve `scikit-learn` pracuje namísto predikcí tříd s predikcemi pravděpodobností tříd od jednotlivých podmodelů (relativní četnosti tříd v listech).
- Tj. např. u binární klasifikace, pokud označíme jako  $\hat{p}_i = P(\hat{Y} = 1 \mid T_i, X = \mathbf{x})$  predikci pravděpodobnosti příslušnosti bodu  $\mathbf{x}$  ke třídě 1 podmodelu  $T_i$ , pak finální predikovaná pravděpodobnost třídy 1 pro náhodný strom je určena průměrem:

$$\hat{p} = \frac{1}{n} \sum_{i=1}^n \hat{p}_i$$

- Finální predikce vysvětlované proměnné  $Y$  je pak určena obvyklým porovnáním této pravděpodobnosti s číslem 1/2 jako

$$\hat{Y} = \begin{cases} 1 & \text{pro } \hat{p} > 0.5 \\ 0 & \text{jinak} \end{cases}$$

## 8.7 Hyperparametry náhodných lesů

- Základními hyperparametry metod `RandomForestClassifier` a `RandomForestRegressor` v `scikit-learn` jsou:
  - `n_estimators`: Určuje počet stromů v náhodném lese.
  - `max_depth`: Určuje maximální hloubku stromů v lese. **Rozdíl oproti boostingu:** Na rozdíl od boostingu (kde se používají mělké stromy jako weak learners), náhodné lesy používají hluboké stromy s nízkým biasem a redukují jejich varianci průměrováním.
  - `max_features`: Počet (náhodně vybraných) príznaků, ze kterých si hladový algoritmus vybírá ten, podle kterého bude v aktuálním kroku data "větvit". Defaultně se volí hodnota  $\sqrt{p}$ , tj. odmocnina počtu príznaků.
  - Je možné nastavovat i další obvyklé hyperparametry rozhodovacích stromů jakožto podmodelů:
    - \* `min_sample_split`: Minimální počet dat v uzlu nutných k rozdelení uzlu.
    - \* `min_sample_leaf`: Minimální počet dat, která musí být v listu.
    - \* `criterion`: Funkce použitá při konstrukci stromu (entropie, gini index).

## 8.8 Poznámky k náhodným lesům

- U baggingu, který skládá rozhodnutí z více podmodelů, je vhodné, aby jednotlivé podmodely nebyly stejné, ale naopak co **nejpestřejší**.
- Toho se docílí nějakým druhem randomizace; v případě náhodných lesů je tento náhodný prvek daný bootstrap metodou pro generování jednotlivých trénovacích datasetů a také hodnotou parametru `max_features`.
- Jak víme, rozhodovací stromy jsou velice citlivé na změny v trénovacích datech (mají velký rozptyl), a tak často stačí odlišnost datasetů daná bootstrapem, abychom získávali velice odlišné rozhodovací stromy.
- Tím, že při baggingu počítáme průměry predikcí těchto rozdílných podmodelů, snažíme se **redukovat rozptyl** (a povětšinou se to úspěšně daří).
- Přestože mohou být jednotlivé stromy samy o sobě slabé modely (podmodelům v ensemble metodách se proto často říká **weak learners**), jejich kolektivní rozhodování dává až překvapivě dobré výsledky.

- Náhodné lesy jsou na rozdíl od rozhodovacích stromů **velice robustní a poměrně odolné vůči přeúčení**. Bohužel ale částečně ztrácejí jejich jednoduchost a snadnou interpretovatelnost.
- U ensemble metod chceme, aby jednotlivé modely (zde stromy) nebyly stejné, ale co nejrůznorodější — toho lze docílit pomocí randomizace — u náhodných lesů toho docílíme pomocí náhodné metody bootstrap a pomocí hodnoty `max_features` (maximální počet features, se kterými každý strom pracuje).

## 8.9 Boosting obecně

**Otázka ke zkoušce 10 (Ensemble metody: bagging, boosting a AdaBoost):** Rozdíl mezi baggingem a boostingem. Popis metody AdaBoost.

- Základní myšlenka boostingu je podobná jako u baggingu: budujeme více modelů (např. rozhodovacích stromů) a jejich finální rozhodnutí je (váženou) kompozicí rozhodnutí jednotlivých modelů.
- **Na rozdíl od baggingu** ale při boostingu nejsou tyto modely nezávislé, ale jsou **seřazeny a každý další je ovlivněn těmi předchozími**.
- Tento vliv je při AdaBoostu realizován pomocí **vah datových bodů**: Při konstrukci  $m$ -tého stromu je zvýšena váha těm bodům, které předchozí  $(m-1)$ -tý strom klasifikoval špatně.
- Takovýmito změnami vah je zajištěno, že se další model soustředí více na ty datové body, se kterými si předchozí modely neporadily.
- Přibližně takto funguje boosting obecně, my si dále ukážeme konkrétní implementaci této myšlenky známou jako algoritmus **AdaBoost** [Freund, Schapire (1997)].

## 8.10 AdaBoost (Adaptive Boosting): popis algoritmu

- Na začátku máme dataset  $\mathcal{D}$  s  $N$  datovými body.
- Pro jednoduchost opět uvažujeme binární klasifikaci. Existují ale i modifikace algoritmu pro více než dvě třídy a i pro regresi.
- Počet zkonstruovaných stromů je zadán uživatelem v hyperparametru `n_estimators`.
- **AdaBoost algoritmus:**

1. Nastavme váhy rovnoměrně, tedy  $w_i = \frac{1}{N}$  a položme  $m = 1$ .
2. Pokud  $m \leq \text{n\_estimators}$ , naučme strom  $T^{(m)}$  na datech  $\mathcal{D}$  s vahami  $w_i$ .
3. Do proměnné  $e^{(m)}$  uložme součet vah těch bodů z  $\mathcal{D}$ , které jsou špatně klasifikované stromem  $T^{(m)}$ .
4. Pokud je  $e^{(m)} = 0$  skončeme, všechna data jsou klasifikována správně.
5. Položme

$$\alpha^{(m)} = \text{learning\_rate} \cdot \log \frac{1 - e^{(m)}}{e^{(m)}},$$

kde `learning_rate` je hyperparametr zadaný uživatelem; používá se ke zpomalení trénování a k zabránění přeúčení (je-li menší než jedna).

6. Pro stromem  $T^{(m)}$  špatně klasifikované body nastavme nové váhy

$$w_i \leftarrow w_i \exp(\alpha^{(m)}).$$

7. Znormalizujme váhy tak, aby jejich součet byl jedna.
8. Zvětšíme  $m$  o jedna a vrátme se do bodu 2.

- Výsledkem algoritmu je tedy až `n_estimators` rozhodovacích stromů  $T^{(1)}, T^{(2)}, \dots$

## 8.11 AdaBoost: predikce

- Abychom určili rozhodnutí tohoto modelu pro nějaký datový bod  $\mathbf{x}$ , postupujeme následovně:
  1. Každému stromu  $T^{(m)}$  přiřadíme váhu danou číslem  $\alpha^{(m)}$  z kroku 5 algoritmu.
  2. Sečti váhy  $\alpha^{(m)}$  všech stromů, které pro  $\mathbf{x}$  predikují  $Y = 1$  a to samé udělej pro stromy predikující  $Y = 0$ .
  3. Rozhodni se pro tu z možností, pro kterou je součet vah vyšší.

## 8.12 Poznámky k AdaBoostu

- Verze algoritmu pro „více než binární“ klasifikaci se nazývá **AdaBoost-SAMME** [Zhu, Rosset, Zou, Hastie (2006)]. Tato verze je implementována v `sklearn`.
- Varianta pro regresi se zase nazývá **AdaBoost.R2** [Drucker (1997)].
- AdaBoost nemusí nutně používat rozhodovací stromy; je možné použít jakýkoli model, který umí pracovat s parametrem `sample_weight`.
- V `sklearn` implementaci jsou rozhodovací stromy (hloubky 3) výchozí volba (parametr `base_estimator`).
- Parametr `learning_rate` je obvykle zaváděn u většiny ensemble metod spadajících do kategorie Boostingu.
- Jedná se o tzv. **regularizaci**: čím je `learning_rate` nižší, tím je model odolnější vůči přeúčení. Nevýhodou je, že je pak obvykle nutné zvýšit počet stromů (parametr `n_estimators`).
- Při boostingu dochází k postupnému korigování chyb předchozích modelů, což vede na **redukci vychýlení (bias)**. Podmodely se typicky konstruují jako **slabé modely (weak learners)** — tj. např. jako poměrně mělké stromy.

## 8.13 Zajímavá videa na YouTube

- [AdaBoost, Clearly Explained \(StatQuest with Josh Starmer\)](#) [21 min]

## 9 Evaluace modelů: metriky

**Otzávka ke zkoušce 11 (Evaluace modelů. Metriky):** Vyhodnocovací metriky regrese a klasifikace (odvozené z matice záměn, ROC, AUC).

## 9.1 Úvod k evaluaci modelů

- Obvykle existuje více kandidátů na finální model a je potřeba vybrat ten nejlepší. K tomu je nutné umět kvantifikovat kvalitu modelu, tedy jeho výkonnost.
- Metodu výpočtu výkonnosti volíme podle cíle (někdy je prioritou nízká chybovost obecně, jindy spíše nižší konkrétní typ chyby, nebo žádné velké chyby ale s možností malých chybovití, atd.).
- **Evaluace modelů:**
  - Hlavní výzvou strojového učení je, aby natrénovaný model dobře fungoval i na nových vstupech, které nikdy neviděl — **schopnost generalizace**.
  - Zaměřuje se na měření výkonnosti modelů supervizovaného učení.

## 9.2 Ztrátová funkce

- Model predikuje  $\hat{Y} = \hat{Y}(X)$ , které je funkcí  $X$ .
- Chybu predikce  $Y$  pomocí  $\hat{Y}$  měříme pomocí **ztrátové funkce** (angl. loss function)  $L$ .
- U **binární klasifikace** model často odhaduje pravděpodobnost:

$$\hat{p} = \hat{P}(Y = 1 \mid X = \mathbf{x})$$

- Běžná ztrátová funkce je **binary cross-entropy loss**:

$$L(Y, \hat{p}) = -Y \log \hat{p} - (1 - Y) \log(1 - \hat{p})$$

## 9.3 Metriky pro regresi

- **MSE** (Mean Squared Error) — nejobvyklejší volba kvadratické chyby jako ztrátové funkce vede na evaluaci pomocí střední kvadratické chyby. Tato míra penalizuje především velké odchylky a je velmi citlivá na odlehlé hodnoty (angl. outliers):

$$MSE = \frac{1}{N} \sum_{i=1}^N (Y_i - \hat{Y}_i)^2$$

- **RMSE** (Root Mean Squared Error) — nelineárně přeškálované  $MSE$  — má stejné vlastnosti jako  $MSE$ , ale je ve stejných jednotkách jako vysvětlovaná proměnná:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N (Y_i - \hat{Y}_i)^2}$$

- **RMSLE** (Root Mean Squared Logarithmic Error) — pro nezáporné hodnoty vysvětlované proměnné, soustředí se na relativní míru odchylek — pro malé hodnoty řeší i malé odchylky, pro velké hodnoty pouze ty velké, méně citlivá na odlehlé hodnoty:

$$RMSLE = \sqrt{\frac{1}{N} \sum_{i=1}^N (\log Y_i - \log \hat{Y}_i)^2}$$

- **MAE** (Mean Absolute Error) — odchylky se skládají lineárně:

$$MAE = \frac{1}{N} \sum_{i=1}^N |Y_i - \hat{Y}_i|$$

- **R<sup>2</sup>** (koeficient determinace) — vyjadřuje, jaký podíl variability cílové proměnné model vysvětluje. Hodnota  $R^2 = 1$  znamená perfektní predikci,  $R^2 = 0$  znamená, že model nevysvětluje žádnou variabilitu (model je stejně dobrý jako průměr):

$$R^2 = 1 - \frac{\sum_{i=1}^N (Y_i - \hat{Y}_i)^2}{\sum_{i=1}^N (Y_i - \bar{Y})^2}$$

## 9.4 Metriky pro klasifikaci

- Při klasifikaci se chyba měřená pomocí ztrátové funkce příliš nehodí.
- Nejčastěji se vyhodnocení klasifikačních modelů provádí od měr odvozených z **matice záměn** (angl. Confusion Matrix) — matice četností různých predikovaných hodnot  $\hat{Y}_i$  proti různým skutečným hodnotám  $Y_i$ .

### 9.4.1 Matice záměn (Confusion Matrix)

- **Základní pojmy pro binární klasifikaci:**
  - **TP** (True Positive) — model správně predikoval  $Y = 1$
  - **FP** (False Positive) — model špatně predikoval  $Y = 0$  jako  $Y = 1$
  - **FN** (False Negative) — model špatně predikoval  $Y = 1$  jako  $Y = 0$
  - **TN** (True Negative) — model správně predikoval  $Y = 0$

|          |                 | Skutečnost      |                         | $\sum$                |
|----------|-----------------|-----------------|-------------------------|-----------------------|
|          |                 | $Y = 1$         |                         |                       |
| Predikce | $\hat{Y} = 1$   | TP              | FP                      | $\hat{N}_+ = TP + FP$ |
|          | $\hat{Y} = 0$   | FN              | TN                      | $\hat{N}_- = FN + TN$ |
| $\sum$   | $N_+ = TP + FN$ | $N_- = FP + TN$ | $N = TP + FP + FN + TN$ |                       |

Tabulka 8: Matice záměn (angl. confusion matrix) pro binární klasifikaci.

#### 9.4.2 Odvozené metriky z matice záměn

- **TPR** (True Positive Rate) — sensitivita neboli **recall** nebo hit rate:

$$TPR = \frac{TP}{N_+} = \frac{TP}{TP + FN}$$

Udává, jakou část skutečně pozitivních případů model správně identifikoval.

- **FPR** (False Positive Rate) — false alarm rate nebo type I error rate:

$$FPR = \frac{FP}{N_-} = \frac{FP}{FP + TN}$$

Udává, jakou část skutečně negativních případů model chybně klasifikoval jako pozitivní.

- **FNR** (False Negative Rate) — miss rate nebo type II error rate:

$$FNR = \frac{FN}{N_+} = \frac{FN}{TP + FN}$$

Udává, jakou část skutečně pozitivních případů model chybně klasifikoval jako negativní.

- **TNR** (True Negative Rate) — specificita nebo selektivita:

$$TNR = \frac{TN}{N_-} = \frac{TN}{FP + TN}$$

Udává, jakou část skutečně negativních případů model správně identifikoval.

- **PPV** (Positive Predictive Value) — **precision** nebo přesnost:

$$PPV = \frac{TP}{\hat{N}_+} = \frac{TP}{TP + FP}$$

Udává, jaká část predikovaných pozitivních případů je skutečně pozitivních.

|               |                        | $\mathbf{Y} = 1$       | $\mathbf{Y} = 0$ |
|---------------|------------------------|------------------------|------------------|
| $\hat{Y} = 1$ | $TPR = \frac{TP}{N_+}$ | $FPR = \frac{FP}{N_-}$ |                  |
|               | $FNR = \frac{FN}{N_+}$ | $TNR = \frac{TN}{N_-}$ |                  |
| $\hat{Y} = 0$ |                        |                        |                  |

Tabulka 9: Míry odvozené z matice záměn vyjádřené jako podíly.

- **ACC** (Accuracy) — přesnost:

$$ACC = \frac{TP + TN}{N} = \frac{TP + TN}{TP + FP + FN + TN}$$

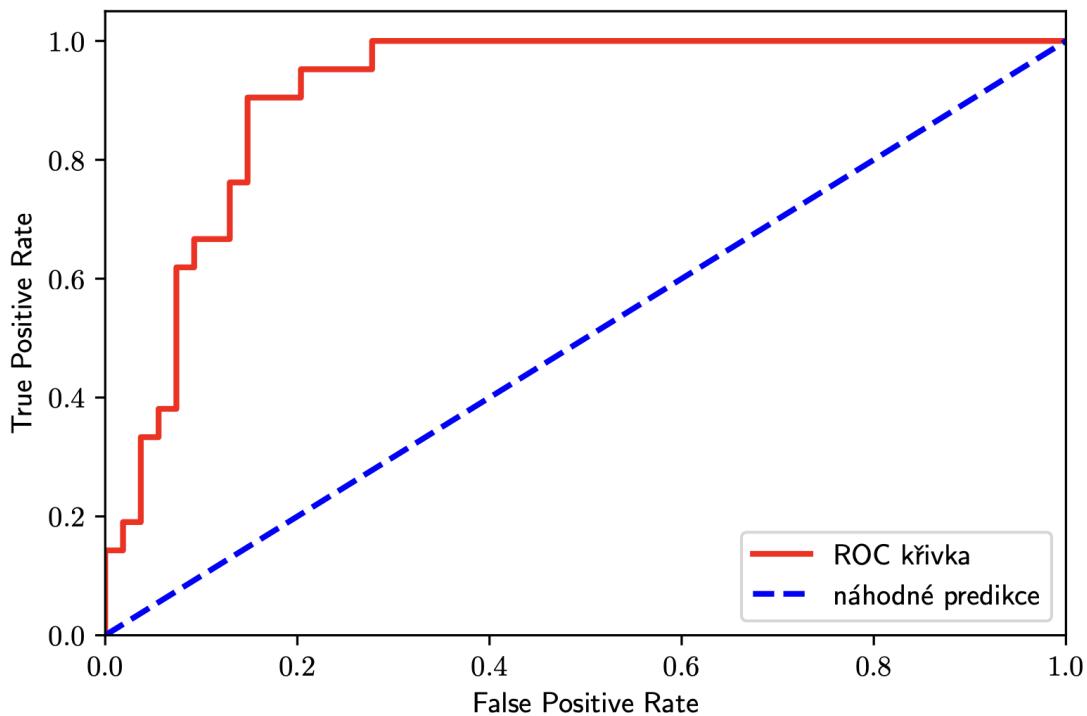
Nejpoužívanější míra (pozor: pokud je dataset nevyvážený, např. 99.9% zdravých a 0.1% nemocných, model predikující vždy zdravý bude mít dobrou přesnost, i když je model k ničemu).

- **F1 score** — harmonický průměr precision a recall — řeší předchozí problém:

$$F_1 = 2 \frac{PPV \cdot TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$$

## 9.5 ROC křivka

- Při predikci používáme jistou hranici  $\tau$  (angl. threshold), podle které určujeme výsledek predikce. Pokud je pravděpodobnost, že vysvětlovaná proměnná záznamu je 1 větší než  $\tau$ , pak predikujeme 1.
- Hodnotu  $\tau$  jsme dotedy nejčastěji brali jako 1/2.
- Chování TPR a FPR v závislosti na  $\tau$ :
  - Pro různé hodnoty prahu  $\tau$  dostáváme různé hodnoty TPR a FPR.
  - Snížením prahu  $\tau$  zvýšíme TPR (zachytíme více pozitivních případů), ale současně zvýšíme i FPR (více falešných poplachů).
  - Zvýšením prahu  $\tau$  snížíme FPR, ale současně snížíme i TPR.
- **ROC křivka** (Receiver Operating Characteristic curve) zobrazuje závislost TPR na FPR pro různé hodnoty prahu  $\tau$ .
- Pro dobrý model bude graf strmě stoupat k levému hornímu rohu a poté již velmi pomalu do pravého horního rohu.



Obrázek 13: ROC křivka — závislost TPR na FPR

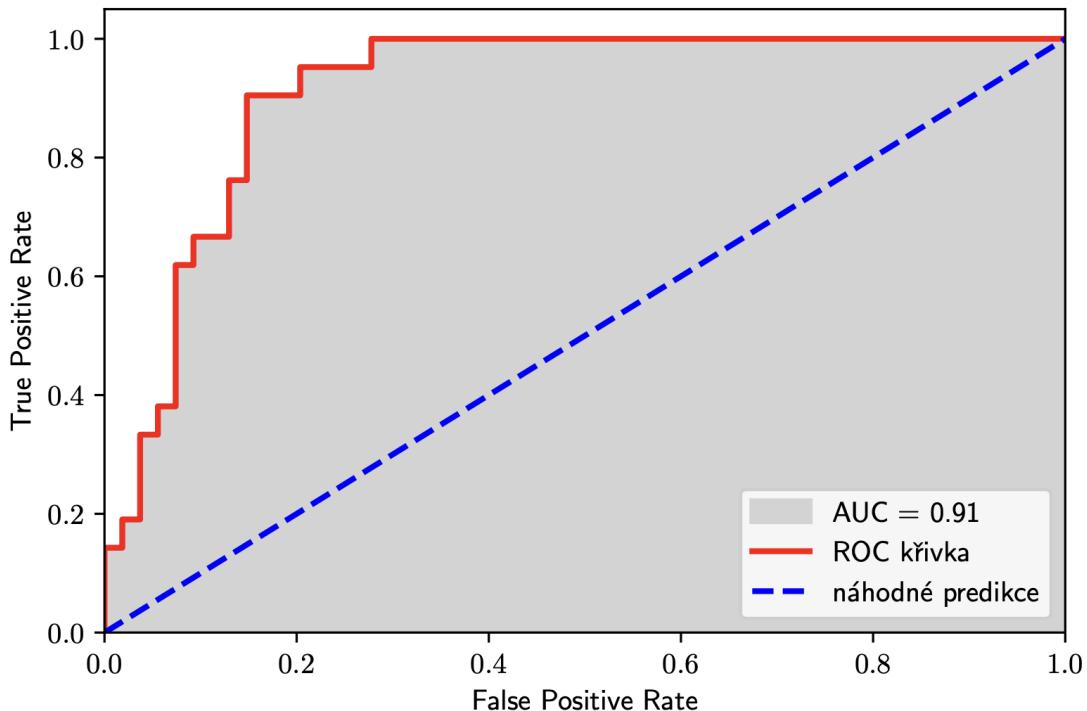
- Interpretace:

- Červená křivka = ROC křivka naučeného modelu
- Modrá čerchovaná přímka = náhodné predikce (model bez predikční síly)
- Čím více se ROC křivka blíží k levému hornímu rohu, tím lepší je model
- Ideální model by měl TPR = 1 a FPR = 0 (levý horní roh)

## 9.6 AUC (Area Under Curve)

- **AUC** je plocha pod ROC křivkou (area under curve).
- Model s náhodnými predikcemi bude mít v průměru plochu pod křivkou 0.5.

- AUC dokonalého modelu bude rovno 1.
- Většina modelů je mezi 0.5 a 1.



Obrázek 14: Plocha pod ROC křívkou ( $AUC = 0.91$ )

- **AUC jako metrika:**

- $AUC = 0.5$ : Model je stejně dobrý jako náhodné hádání.
- $0.5 < AUC < 1.0$ : Model je lepší než náhodné hádání, ale není bezchybný v predikcích.
- $AUC = 1.0$ : Perfektní model.

- **Výhody AUC:**

- Nezávislá na volbě prahu  $\tau$
- Vhodná pro nevyvážené datasety
- Poskytuje celkový pohled na výkonnost modelu napříč všemi prahy

## 10 Evaluace modelů: testovací chyba a její odhad

**Otzávka ke zkoušce 12 (Evaluace modelů):** Testovací chyba a její odhad. Evaluační scénáře - trénovací, validační a testovací množina versus křížová validace.

### 10.1 Vyhodnocovací scénáře

- Chceme-li natrénovat model a pak odhadnout jeho testovací chybu, musíme mít k dispozici trénovací data a nezávislá testovací data.
- Typicky máme více kandidátů na finální model a snažíme se vybrat nejlepší. To znamená najít hodnoty hyperparametrů tak, aby testovací chyba finálně vybraného modelu byla co nejmenší.
- **Z pohledu evaluace máme dva úkoly:**

- **Výběr modelu** — odhadnout výkonnost různých modelů a vybrat nejlepší.
- **Ohodnocení modelu** — odhadnout testovací chybu finálního modelu.
- Protože výběr modelu vlastně spadá do procesu trénování (vybereme model, který se nejlépe přizpůsobí datům), nesmíme použít stejná data pro výběr finálního modelu i pro ohodnocení tohoto finálního modelu. Data rozdělíme.

## 10.2 Trénovací, validační a testovací množina

- Když máme dostatek dat, rozdělíme vstupní data na tři části: **trénovací, validační a testovací**.



Obrázek 15: Rozdělení dat na trénovací, validační a testovací množinu

- **Trénovací část** — použijeme pro trénování konkrétních modelů se zafixovanými hyperparametry.
- **Validační část** — použijeme k ohodnocení daného modelu a porovnání s ostatními modely. Finální model vybereme na základě validační množiny.
- **Testovací část** — použijeme až v závěrečné fázi, když už máme vybraný a natrénuovaný finální model. Tím získáme odhad testovací chyby, kterou můžeme očekávat na nových datech. Tento způsob práce s testovací množinou se nazývá **hold-out**.
- Trénovací, validační a testovací data by měla pocházet ze stejného rozdělení.
- V případě chronologických dat (ceny na burze) není vhodné předem permutovat data — testovací (a případně i validační) data by měla správně reflektovat chronologické řazení.

## 10.3 Křížová validace

- Když nemáme dostatek dat, nebývá rozumné dělit je na trénovací, validační a testovací část.
- Oddělíme si testovací data, ale trénovací a validační již dělit nebudeme.
- **$k$ -násobná křížová validace** (angl.  $k$ -fold cross-validation):

- Trénovací data  $\mathcal{D}$  náhodně rozdělíme na  $k$  podobně velkých částí  $\mathcal{D}_1, \dots, \mathcal{D}_k$ .
- Pro každé  $j = 1, \dots, k$  model s danými hodnotami hyperparametrů natrénujeme na datech z množiny  $(\bigcup_{i=1}^k \mathcal{D}_i) \setminus \mathcal{D}_j$ .
- Na množině  $\mathcal{D}_j$  odhadneme jeho chybu jako  $e_j$ .
- Nakonec vrátíme průměrnou **cross-validační chybou**:

$$\hat{e} = \frac{1}{k} \sum_{i=1}^k e_i$$

- Tento proces zopakujeme pro všechny zkoumané hodnoty hyperparametrů a na závěr vybereme jako nejlepší volbu ty hodnoty, které vedly k nejmenší cross-validační chybě.
- Hodnota  $k$  se typicky volí 5 až 10.



Obrázek 16: Příklad 5-násobné křížové validace

- **Poznámky:**

- Může být neúnosně výpočetně náročná.
- Pro fixní model je cross-validační chyba odhadem očekávané testovací chyby  $Err$ , nikoliv testovací chyby  $Err_{\mathcal{D}}$ .
- V případě, kdy máme opravdu málo dat a nechceme ani oddělovat testovací množinu, můžeme namísto toho udělat dvoustupňovou validaci — vnitřní křížovou validaci používáme na výběr nejlepšího modelu a vnější na odhad očekávané chyby. Výsledná vnější cross-validační chyba ale odpovídá chybě celé procedury pro výběr nejlepšího modelu, nikoliv chybě konkrétního modelu.

## 10.4 Trénovací chyba

- Při učení se snažíme minimalizovat chybu predikce měřenou pomocí průměrné hodnoty ztrátové funkce  $L$  na trénovacích datech, kde trénovací množina je  $N$  dvojic  $(Y_i, \mathbf{x}_i)$ :

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N L(Y_i, \hat{Y}(\mathbf{x}_i))$$

- Tato průměrná hodnota se nazývá **trénovací chyba** (angl. training error) a občas se značí jako  $err_{train}$ . Někdy se jí také říká trénovací ztráta (angl. training loss).

## 10.5 Učení modelu

- Během procesu učení modelu se zafixovanými hyperparametry se snažíme najít hodnoty parametrů, které minimalizují trénovací chybu.
- Jakmile je model natrénován, zajímá nás jeho **schopnost generalizace** — jak dobře funguje na nových datech.

## 10.6 Testovací chyba

- **Testovací chyba** (angl. test error) neboli **generalizační chyba** (angl. generalization error) je střední hodnota chyby na novém vstupu  $X$  podmíněná danými trénovacími daty:

$$Err_{\mathcal{D}} = \mathbb{E} [L(Y, \hat{Y}(X)) \mid \mathcal{D}]$$

kde  $\mathcal{D} = \{(Y_1, \mathbf{x}_1), \dots, (Y_N, \mathbf{x}_N)\}$  značí trénovací data.

- Testovací chybu můžeme odhadnout výběrovým průměrem:

$$err_{test} = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} L(Y_i, \hat{Y}(\mathbf{x}_i))$$

změřeným na testovacích datech  $(Y_1, \mathbf{x}_1), \dots, (Y_{N_{test}}, \mathbf{x}_{N_{test}})$ , které byla získána nezávisle na trénovacích datech  $\mathcal{D}$ .

- Poznamenejme, že  $err_{test}$  podmíněno trénovacími daty je nestranný odhad  $Err_{\mathcal{D}}$ , tj.:

$$\mathbb{E} [err_{test} \mid \mathcal{D}] = Err_{\mathcal{D}}$$

- **Očekávaná testovací chyba** neboli **očekávaná chyba predikce** (angl. expected test error, expected prediction error) je definována jako střední hodnota testovací chyby vzhledem k náhodnému výběru trénovací množiny:

$$Err = \mathbb{E} [Err_{\mathcal{D}}] = \mathbb{E} [L(Y, \hat{Y}(X))]$$

# 11 Výběr příznaků (Feature Selection)

**Otázka ke zkoušce 13 (Výběr příznaků):** Základní metody výběru příznaků (filtrační, obalové, vestavěné), Lasso.

## 11.1 Úvod do výběru příznaků

- **Výběr příznaků** (angl. feature selection nebo také subset selection) je proces, při kterém se ze všech dostupných příznaků vybírá jistá podmnožina, kterou se použije pro trénování a predikci modelu.
- Problematika výběru příznaků spadá do části předzpracování dat.
- Z jistého pohledu se jedná o podoblast **redukce dimenzionality** (angl. dimensionality reduction). Do té ale spadají především metody, které napočítávají (lineárně nebo nelineárně) kompletne nové příznaky. Proto se redukce dimenzionality často uvádí jako separátní oblast.

- **Účel výběru příznaků:**

- Zahodením nerelevantních a redundantních příznaků lze významně zlepšit schopnost generalizace modelu.
- Pomáhá s prokletím dimenzionality — pro některé modely je velká dimenze prostoru příznaků problematická (např. kNN).
- Zlepšuje vysvětlitelnost modelu — u modelů, které využívají menší počet příznaků, bývá jednodušší porozumět tomu, na základě čeho se rozhodují.
- Snižuje výpočetní nároky pro trénování a použití výsledného modelu.

- **Poznámky k nerelevantním a redundantním příznakům:**

- Příznaky, které jsou nerelevantní a nesouvisí s vysvětlovanou proměnnou, škodí typicky i modelům, které by si s nimi teoreticky dokázaly poradit — jako např. lineární regrese (může tam použít koeficient 0) nebo rozhodovací strom (nepoužije tento příznak do žádného dělícího kritéria).
- Redundantní příznaky jsou takové, které nesou stejnou informaci. Kromě toho, že zbytečně zvyšují dimenzi, mohou některým modelům vyloženě škodit, jako např. problém kolinearity u lineární regrese.

## 11.2 Filtrační metody

- **Filtrační metody** (angl. filter methods) mohou být supervizované i nesupervizované. Typicky zkoumají, jak hodně informace může být v daném příznaku a případně jak ta informace souvisí s vysvětlovanou proměnnou.
- **Nesupervizované filtrační metody** (bez využití vysvětlované proměnné):
  - Vyhodit příznaky, které mají příliš nízký rozptyl a jsou tedy téměř konstantní.
  - Vyhodit příznaky, které mají příliš chybějících hodnot.
  - Vyhodit některé z příznaků, které spolu hodně korelují a jsou tedy redundantní.
- **Supervizované filtrační metody** (s využitím vysvětlované proměnné):
  - Vyhodit příznaky, které mají nízkou korelací s vysvětlovanou proměnnou.
  - U binárních příznaků rozdělit vysvětlovanou na dvě populace a udělat test hypotézy o rovnosti středních hodnot (dvouvýběrový t-test). Pokud vyjde velká p-hodnota, lze tento příznak vyhodit.
  - U diskrétních příznaků i diskrétní vysvětlované proměnné udělat test hypotézy o nezávislosti těchto dvou diskrétních veličin (např. chí-kvadrát test nezávislosti). Pokud vyjde velká p-hodnota, je tento příznak kandidátem na vyhození.

### 11.3 Obalové metody

- **Obalové metody** (angl. wrapper methods) využívají nějaký pomocný model pro ohodnocování podmnožin příznaků. Cílem je vybrat takovou podmnožinu, pro kterou je výkonnost modelu největší.
- Pro každou kandidátní podmnožinu se model natrénuje a změří jeho výkonnost na validační množině (nebo křížovou validaci). Jako finální podmnožina příznaků je použita taková, pro kterou je model nejvýkonnéjší.
- **Výhody:** Preferují podmnožiny příznaků, které jsou pro daný model výhodné.
- **Nevýhody:**
  - Může snadno dojít k přeučení.
  - Taktiž vybrané podmnožiny nemusí být vhodné pro jiné modely (např. pro ten finální, který se má použít).
  - Vysoká výpočetní náročnost — i při použití jednoduchého pomocného modelu (jako např. lineární regrese) může být výpočetní čas značný.
- **Nejpoužívanější obalové metody:**
  - **Dopředný výběr** (angl. forward selection) — začíná se s prázdnou množinou příznaků a postupně se po jednom přidávají vždy tak, aby ten jeden přidaný nejvíce zlepšil výkonnost modelu v dané iteraci. Končí se, když je dosažen požadovaný počet příznaků, případně když už nedochází ke zlepšování výkonnosti.
  - **Zpětný výběr** (angl. backward selection) — začíná se se všemi příznaky a postupně se po jednom odebírají tak, aby ten odebraný příznak vedl na nejvýkonnéjší model v dané iteraci. Končí se, když je dosažen požadovaný počet příznaků, případně když už nedochází ke zlepšování výkonnosti.
  - **Rekurzivní odebíráni příznaků** (angl. recursive feature elimination) — probíhá podobně jako zpětný výběr, ale model se využívá trochu jiným způsobem. Konkrétně se k vybíráni příznaků používá vnitřní ohodnocení důležitosti jednotlivých příznaků modelem. U lineární (logistické) regrese to je například absolutní hodnota koeficientu u daného příznaku. U rozhodovacího stromu to může být informační zisk daného příznaku.
- **Implementace v sklearn:**
  - Dopředný i zpětný výběr: `sklearn.feature_selection.SequentialFeatureSelector`
  - Rekurzivní odebíráni: `sklearn.feature_selection.RFE`

### 11.4 Vestavěné metody

- **Vestavěné metody** (angl. embedded methods) využívají model, který se trénuje pouze jednou na celých datech a při tom implicitně provede výběr příznaků.
- Tento implicitní výběr se projeví tak, že se model naučí některé příznaky vůbec nevyužívat — např. u lineární regrese jsou příslušné koeficienty odhadnuty jako 0.
- Modelem, který je nejpoužívanější, je  $L_1$  regularizovaná lineární regrese (**Lasso**). V případě klasifikace to je pak  $L_1$  regularizovaná logistická regrese.
- Jako vybraná podmnožina příznaků se vezmou příznaky, u kterých jsou příslušné koeficienty nenulové.

### 11.5 Lasso regrese

- **Lasso** (zkr. angl. Least Absolute Shrinkage and Selection Operator) [Tibshirani (1996)] nebo také  $L_1$  regularizace zavádí penalizační člen úměrný součtu absolutních hodnot složek vektoru koeficientů  $\mathbf{w}$  s vynescháním interceptu.
- Minimalizuje se **regularizovaný reziduální součet čtverců**:

$$RSS_{\lambda}^{Lasso}(\mathbf{w}) = \|\mathbf{Y} - \mathbf{X}\mathbf{w}\|^2 + \lambda \sum_{i=1}^p |w_i|$$

který závisí na parametru  $\lambda \geq 0$ .

- **Vlastnosti Lasso:**

- Pro  $\lambda = 0$  dostáváme  $RSS_0^{Lasso}(\mathbf{w}) = RSS(\mathbf{w})$  a máme tedy obyčejnou metodu nejmenších čtverců.
- Pro  $\lambda > 0$  se v minimu cílí na takové vektory  $\mathbf{w}$ , které mají co nejmenší složky.
- Hodnota  $w_0$  interceptu se nijak nepenalizuje. Jedná se pouze o vertikální posun, který zajišťuje předpoklad  $\mathbb{E}\varepsilon = 0$  modelu a je tedy vhodné ho neomezovat.

- **Rozdíl oproti hřebenové regresi:**

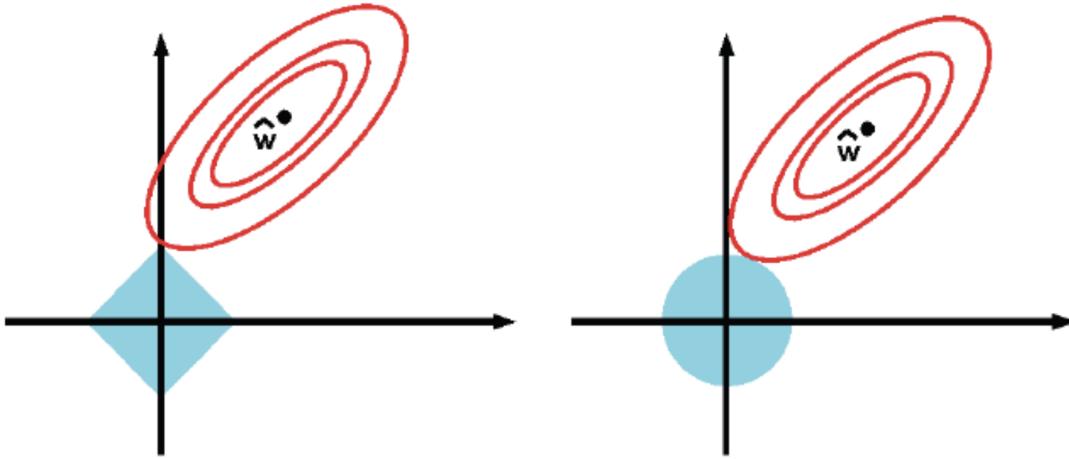
- Na rozdíl od modelu hřebenové regrese není regularizační člen  $\sum_{i=1}^p |w_i|$  diferencovatelný v bodech, kde  $w_j = 0$ .
- V tomto případě není možné nalézt explicitní řešení a existují pouze iterativní metody, které najdou řešení:

$$\hat{\mathbf{w}}_\lambda^{Lasso} = \arg \min_{\mathbf{w}} RSS_\lambda^{Lasso}(\mathbf{w})$$

- **Řídké řešení** (angl. sparse solution):

- Výhoda modelu Lasso je, že řešení je tzv. řídké.
- To znamená, že odhad  $\hat{w}_{\lambda;j}^{Lasso}$  některých složek  $\mathbf{w}$  jsou rovny 0. Samozřejmě tím častěji, čím je  $\lambda$  větší.
- Tím se Lasso zásadně liší od hřebenové regrese, kde tento efekt v podstatě nikdy nenastane.

- **Geometrická interpretace:**



Obrázek 17: Porovnání regularizace Lasso (vlevo) a Ridge (vpravo)

Na obrázku 17 jsou znázorněny vrstevnice funkcí, které se minimalizují u Lasso (vlevo) a hřebenové regrese (vpravo). Na osách jsou jednotlivé složky vektoru vah  $w_1$  a  $w_2$ . Červené elipsy jsou vrstevnice odpovídající neregularizované části  $\|\mathbf{Y} - \mathbf{X}\mathbf{w}\|^2$ , což je v zásadě parabolická jáma s minimem v bodě  $\hat{\mathbf{w}}_{OLS}$ . Tyrkysově je pak znázorněna oblast, která odpovídá regularizačnímu členu (omezení na normu vah).

U Lasso má tato oblast tvar **kosočtverce** (resp. v obecné dimenzi polytop), protože omezení  $\sum_{i=1}^p |w_i| \leq B$  definuje  $L_1$  kouli. U hřebenové regrese má oblast tvar **kruhu** (resp. v obecné dimenzi hyperkoule), protože omezení  $\sum_{i=1}^p w_i^2 \leq B$  definuje  $L_2$  kouli.

U Lasso existuje velká šance, že se vrstevnice RSS dotýká některého z rohů kosočtverce, což znamená, že je jedna ze složek  $\mathbf{w}$  rovna 0. U hřebenové regrese se to samozřejmě také může stát, ale je to velmi nepravděpodobné, protože kruh nemá rohy.

- **Ekvivalentní formulace jako vázaná optimalizace:**

Úloha minimalizace  $RSS_\lambda^{Lasso}(\mathbf{w})$  je ekvivalentní úloze vázané minimalizace:

$$\min_{\mathbf{w}} RSS(\mathbf{w}) = \|\mathbf{Y} - \mathbf{X}\mathbf{w}\|^2 \quad \text{za podmínky} \quad \sum_{i=1}^p |w_i| \leq B, \quad \text{pro nějaké } B > 0.$$

Analogicky pro hřebenovou regresi platí, že minimalizace  $RSS_{\lambda}^{Ridge}(\mathbf{w})$  je ekvivalentní úloze vázané minimalizace  $RSS(\mathbf{w})$  za podmínky  $\sum_{i=1}^p w_i^2 \leq B$ .

- **Porovnání OLS, Ridge a Lasso pro ortonormální příznaky ( $\mathbf{X}^T \mathbf{X} = I$ ):**

- Metoda nejmenších čtverců:  $\hat{\mathbf{w}}_{OLS} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y} = \mathbf{X}^T \mathbf{Y}$

- Hřebenová regrese:

$$\hat{w}_{\lambda;0}^{Ridge} = \hat{w}_0^{OLS} \quad \text{a} \quad \hat{w}_{\lambda;j}^{Ridge} = \frac{1}{1+\lambda} \hat{w}_j^{OLS}$$

- Lasso (soft thresholding):

$$\hat{w}_{\lambda;0}^{Lasso} = \hat{w}_0^{OLS} \quad \text{a} \quad \hat{w}_{\lambda;j}^{Lasso} = \text{sgn}(\hat{w}_j^{OLS}) \cdot \max \left( 0, |\hat{w}_j^{OLS}| - \frac{\lambda}{2} \right)$$

- **Implementace a poznámky:**

- Lasso je v knihovně `sklearn` implementováno pomocí `sklearn.linear_model.Lasso`.
- Při využití pro výběr příznaků se dá použít implementace `sklearn.feature_selection.SelectFromModel`.
- U logistické regrese by se regularizační člen přidal k mínu logaritmu věrohodnostní funkce (k binární relativní entropii) a pak by se prováděla minimalizace.
- Jednou z nevýhod Lasso proti hřebenové regresi je, že v případě kolineárních příznaků má tendenci vybrat pouze některé z nich. To se v některých případech při predikci na nových datech ukazuje jako jistá nevýhoda oproti hřebenové regresi, která typicky využije všechny příznaky.

## 11.6 Elastic Net regrese

- **Elastic Net** je model, který kombinuje oba způsoby regularizace —  $L_1$  (Lasso) i  $L_2$  (Ridge) — a spojuje tak výhody obou přístupů.
- Minimalizuje se:

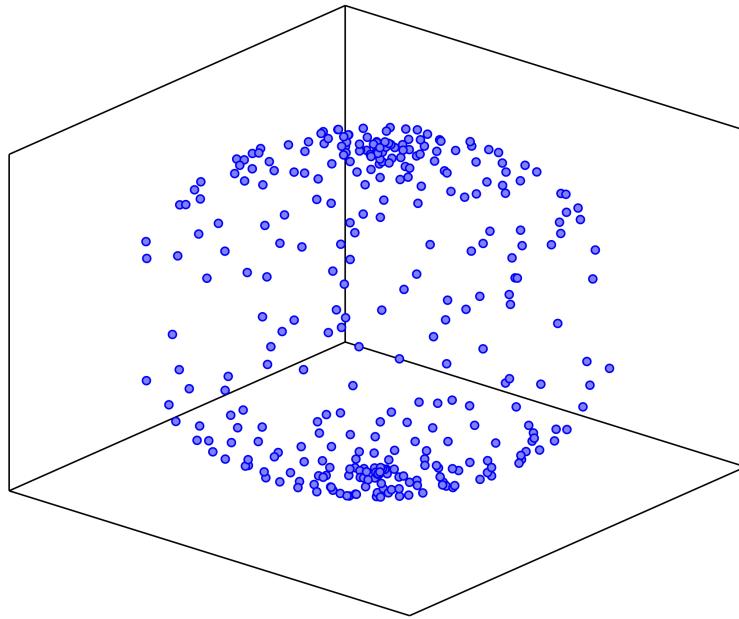
$$RSS_{\lambda_1, \lambda_2}^{ElasticNet}(\mathbf{w}) = \|\mathbf{Y} - \mathbf{X}\mathbf{w}\|^2 + \lambda_1 \sum_{i=1}^p |w_i| + \lambda_2 \sum_{i=1}^p w_i^2$$

## 12 Hierarchické shlukování (Hierarchical Clustering)

**Otzávka ke zkoušce 14 (Hierarchické shlukování):** Cíle nesupervizovaného učení a shlukování. Vzdálenost (metrika): definice a příklady. Aglomerativní algoritmus, měření vzdáleností shluků, dendrogram.

### 12.1 Nesupervizované učení (Unsupervised Learning)

- **Nesupervizované učení** (angl. unsupervised learning) nastává v situaci, kdy data nemáme nikterak označena. Tj. nemáme žádnou veličinu, kterou bychom u trénovacích dat znali a snažili se ji naučit predikovat.
- Cílem nesupervizovaného učení je porozumět struktuře dat pouze na základě jich samotných. To znamená bez nějakého vnějšího vodítka. Proto se nesupervizovanému učení také říká **učení bez učitele**.
- Porozuměním zde typicky rozumíme nalezení co „nejmenších“ oblastí v prostoru příznaků, kde se data vyskytují nejčastěji.
- Obvykle totiž platí, že se naměřená skutečná data nevyskytují v celém prostoru stejně pravděpodobně, ale bývají více či méně lokalizována — tvoří nějaké shluky, vyskytují se v méně-dimenzionálních oblastech atd.
- Porozumění této lokalizaci přináší důležitou informaci o vnitřní struktuře dat!



Obrázek 18: Data rozložená v třírozměrném prostoru na kouli

- Na obrázku 18 jsou znázorněna data rozložená v třírozměrném prostoru. Podaří-li se zjistit, že jsou všechny tyto body na kouli, je možné jejich polohu popsat pomocí sférických souřadnic a získat tak ekvivalentní reprezentaci pomocí dvou spojitých příznaků (tzv. **redukce dimenzionality**). Navíc je možné detektovat, že hustota bodů na pólech je vyšší než hustota bodů na rovníku.
- **Obecný problém nesupervizovaného učení:** Vůbec není jasné, jak bychom měli vyhodnocovat úspěšnost získaného porozumění. To je velký rozdíl oproti supervizovanému učení, kde je možné kvalitu naučeného modelu vyhodnocovat mnoha víceméně rovnocennými způsoby (např. přesností u klasifikace).

## 12.2 Shluková analýza (Cluster Analysis)

- Jednou ze základních metod nesupervizovaného učení je **shlukování** nazývané také **clusterování** (angl. clustering).
- Cílem je roztrídit data do skupin, které se nazývají **shluky**, tak, že platí dva přirozené požadavky:
  - Blízké body budou ve stejném shluku.
  - Vzdálené body budou v různých shluclích.
- Tento popis je ovšem hodně vágní a není vůbec jasné, jak ho zformalizovat. Jedním z problémů je fakt, že tyto intuitivní požadavky obecně nejsou kompatibilní.
- Dalším problémem je již dříve zmínovaná neexistence hodnotícího kritéria kvality nalezeného shlukování.
- V důsledku absence jednoznačnosti tudíž existuje více způsobů formalizace a následně také mnoho různých shlukovacích algoritmů, které mohou na stejných datech dávat velmi rozdílné výsledky.

### 12.2.1 Vzdálenost (metrika)

- Klíčovým pojmem, na kterém stojí shlukování, je **vzdálenost**.
- **Definice:** Vzdálenost nebo také **metrika** na množině  $\mathcal{X}$  je funkce  $d : \mathcal{X} \times \mathcal{X} \rightarrow [0, +\infty)$  taková, že pro každé  $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathcal{X}$  platí:
  1.  $d(\mathbf{x}, \mathbf{y}) \geq 0$ , a  $d(\mathbf{x}, \mathbf{y}) = 0$  právě tehdy když  $\mathbf{x} = \mathbf{y}$  — **pozitivní definitnost**,
  2.  $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$  — **symetrie**,

3.  $d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{z}, \mathbf{y})$  — trojúhelníková nerovnost.

- Dvojice  $(\mathcal{X}, d)$  se potom nazývá **metrický prostor**.

- Poznámky k vlastnostem vzdálenosti:

- Vzdálenost různých bodů je vždy kladná. Nulová je pouze pro stejné body.
  - Vzdálenost bodu  $x$  od bodu  $y$  je stejná jako vzdálenost  $y$  od  $x$ .
  - Přímá vzdálenost mezi dvěma body je vždy menší nebo rovna vzdálenosti přes nějaký třetí bod.

### 12.2.2 Příklady vzdáleností na $\mathbb{R}^p$

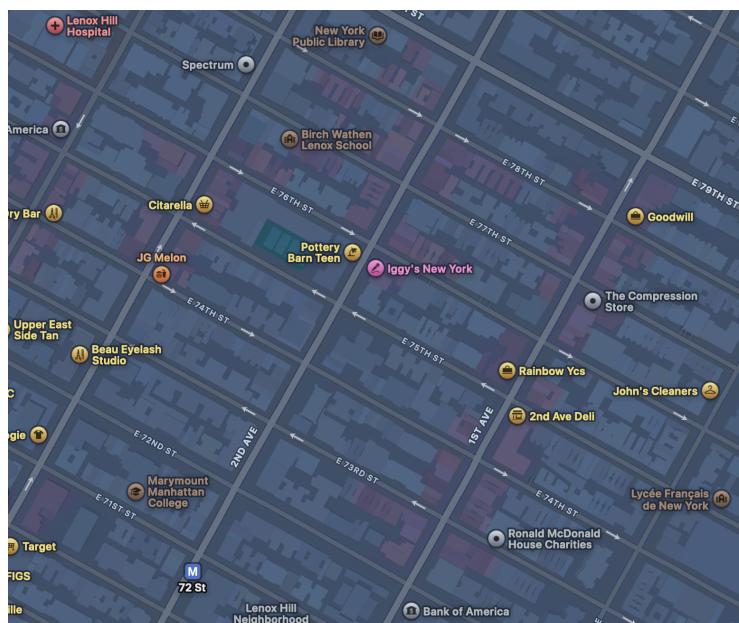
- Euklidovská vzdáenosť (nebo také  $L_2$  vzdáenosť):

$$d_2(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^p (x_i - y_i)^2}$$

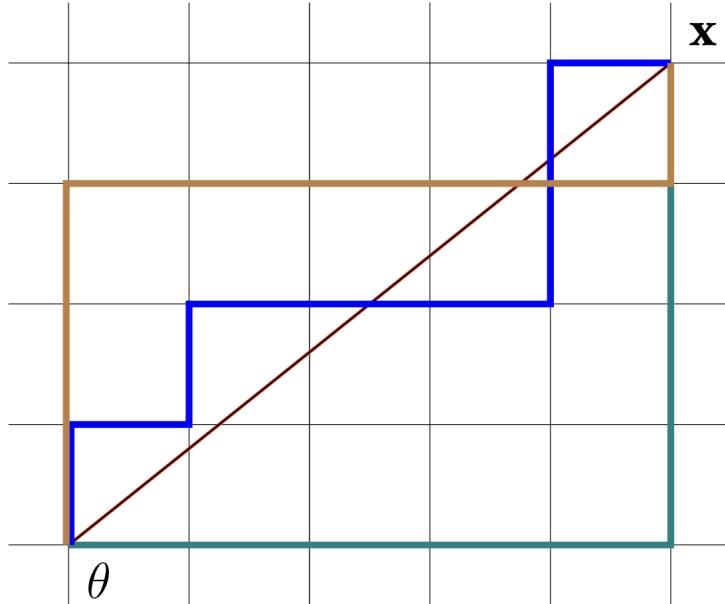
- Manhattanská vzdálenost (nebo také  $L_1$  vzdálenost):

$$d_1(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^p |x_i - y_i|$$

Manhattanská vzdálenost je pojmenována podle ulice Manhattanu v New Yorku, kde je uliční síť pravidelná čtvercová mřížka.



Obrázek 19: Mapa Manhattanu v New Yorku připomíná čtvercovou mřížku. Zdroj: Apple Maps



Obrázek 20: Ve čtvercové mřížce má každá cesta z počátku do  $\mathbf{x}$  délku alespoň  $\|\mathbf{x}\|_1$ . Zde  $\|(5, 4)\|_1 = 9$ .

- **Čebyševova vzdálenost** (nebo také  $L_\infty$  vzdálenost):

$$d_\infty(\mathbf{x}, \mathbf{y}) = \max_i |x_i - y_i|$$

- Příkladem vzdálenosti na množině řetězců je **Levenštejnova vzdálenost** (angl. Levenshtein distance) definovaná jako minimální počet jednoznakových operací (vkládání, mazání, nahrazení), které převedou jeden řetězec na druhý.

### 12.2.3 Vstupy a výstupy shlukování

- **Vstupy:**

- Metrický prostor  $\mathcal{X}$  se vzdáleností  $d$ .
- Množina dat  $\mathcal{D} \subset \mathcal{X}$ .
- Obvykle také požadovaný počet shluků  $k$ .

- **Výstupy:**

- Rozklad množiny dat na jednotlivé shluky. To jest  $\mathcal{C} = (C_1, \dots, C_k)$ , kde  $C_i \subset \mathcal{D}$  pro každé  $i$  a  $C_i \cap C_j = \emptyset$  pro každé  $i \neq j$ , přičemž:

$$\mathcal{D} = \bigcup_{i=1}^k C_i$$

- Bod  $\mathbf{x} \in \mathcal{D}$  je tedy v  $i$ -tém shluku, jestliže  $\mathbf{x} \in C_i$ .
- U hierarchického shlukování může být finálním výstupem **dendrogram** jakožto grafické znázornění hierarchické struktury shlukování.

## 12.3 Hierarchické shlukování

- **Agglomerativní hierarchické shlukování** je přirozený hladový přístup. Označme  $N$  počet prvků množiny dat  $\mathcal{D}$ .
- **Algoritmus:**

1. Na začátku uvažujeme každý bod jako jeden shluk. Tj. máme právě  $N$  shluků.
  2. Nyní opakujeme následující kroky:
    - Najdeme dva shluky, které jsou k sobě nejblíže.
    - Tyto dva shluky spojíme do nového shluku.
  3. Po  $N - 1$  opakování skončíme s jediným velkým shlukem, ve kterém jsou všechny body.
- K tomu, aby bylo možné tento postup provést, je třeba stanovit způsob **měření vzdálenosti dvou shluků**.
  - Má-li být navíc výstupem shlukování, je nutné určit nějaké **zastavovací kritérium**. To bývá nejčastěji počet shluků  $k$ , případně limitní hodnota vzdálenosti shluků, nad kterou už nebudeme shluky spojovat.

### 12.3.1 Měření vzdálenosti shluků

- Vstupem do shlukování je vzdálenost dvojice bodů  $d(\mathbf{x}, \mathbf{y})$ . K měření vzdálenosti dvojcích shluků  $D(A, B)$  se obvykle používá jeden z následujících způsobů:
  - **Metoda nejbližšího souseda** (angl. single linkage) — generuje dlouhé řetězce:
- $$D(A, B) = \min_{\mathbf{x} \in A, \mathbf{y} \in B} d(\mathbf{x}, \mathbf{y})$$
- **Metoda nejvzdálenějšího souseda** (angl. complete linkage) — generuje kompaktní shluky:
- $$D(A, B) = \max_{\mathbf{x} \in A, \mathbf{y} \in B} d(\mathbf{x}, \mathbf{y})$$
- **Párová vzdálenost** (angl. average linkage) — kompromis předchozích dvou:

$$D(A, B) = \frac{1}{|A||B|} \sum_{\mathbf{x} \in A, \mathbf{y} \in B} d(\mathbf{x}, \mathbf{y})$$

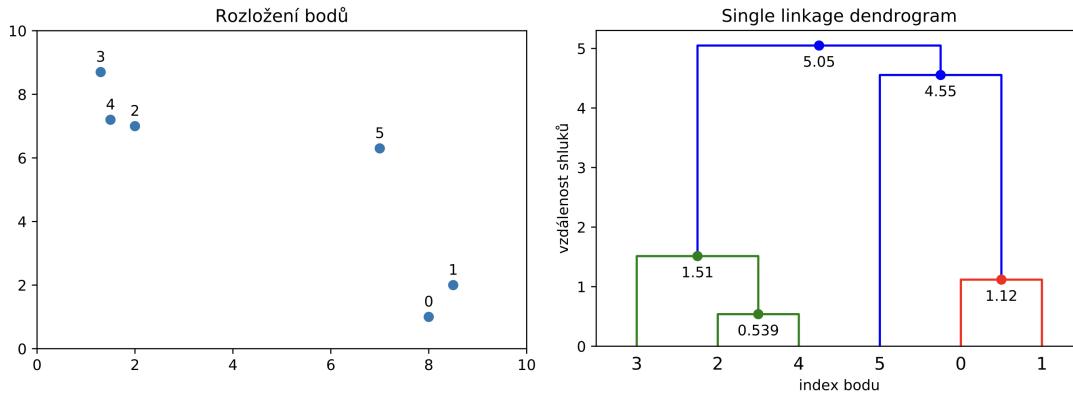
- **Wardova metoda** — v  $\mathbb{R}^p$  velmi účinná, minimalizuje nárůst vnitřního rozptylu:

$$D(A, B) = \sum_{\mathbf{x} \in A \cup B} \|\mathbf{x} - \bar{\mathbf{x}}_{A \cup B}\|^2 - \sum_{\mathbf{x} \in A} \|\mathbf{x} - \bar{\mathbf{x}}_A\|^2 - \sum_{\mathbf{x} \in B} \|\mathbf{x} - \bar{\mathbf{x}}_B\|^2$$

kde  $\bar{\mathbf{x}}_A = \frac{1}{|A|} \sum_{\mathbf{x} \in A} \mathbf{x}$  je geometrický střed množiny  $A$  a  $\bar{\mathbf{x}}_B$ ,  $\bar{\mathbf{x}}_{A \cup B}$  analogicky.

### 12.3.2 Vizualizace pomocí dendrogramu

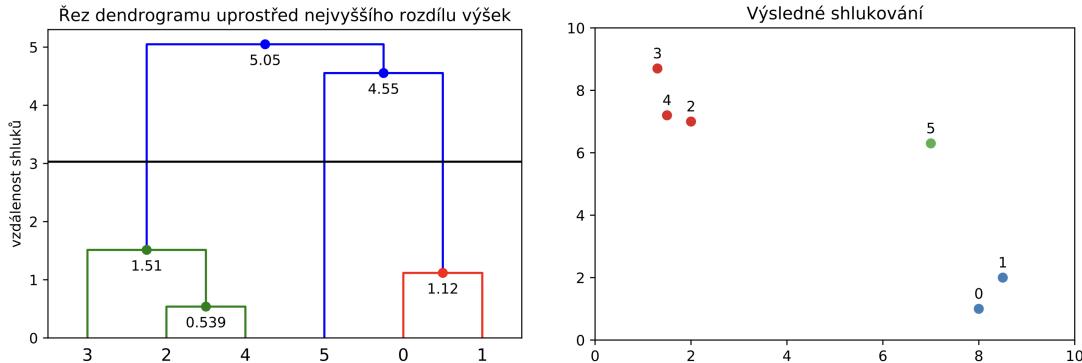
- Celý proces agglomerativního shlukování je možné reprezentovat a graficky znázornit pomocí **dendrogramu**.
- Jedná se o strom, jehož vrcholy představují shluky, které při běhu vznikly.
- V listech jsou počáteční jednoprvkové shluky a kořen reprezentuje finální shluk všech bodů.
- Strom je navíc nakreslený tak, že jsou všechny listy ve výšce 0 a výška ostatních vrcholů odpovídá vzdálenosti podřazených shluků, které se v tomto vrcholu spojily.



Obrázek 21: Rozložení bodů a odpovídající dendrogram (single linkage)

### 12.3.3 Získání shlukování z dendrogramu

- Známe-li požadovaný počet  $k$  shluků, „rozízneme“ dendrogram mezi  $k$ -tým a  $(k - 1)$ -tým nejvyšším vrcholem. Hrany procházející říznutím pak odpovídají výsledným shlukům.
- Nebo je možné stanovit limitní hranici vzdálenosti shluků a rozříznout dendrogram v dané výšce.
- Případně lze určovat místo rozříznutí a tím potažmo i počet shluků pomocí rozdílů výšek sousedních vrcholů.



Obrázek 22: Řez dendrogramu uprostřed nejvyššího rozdílu výšek a výsledné shlukování

### 12.3.4 Poznámky k hierarchickému shlukování

- Ukázali jsme si **agglomerativní** hierarchické shlukování.
- Existuje také **divizní** hierarchické shlukování, při kterém naopak vycházíme z jednoho shluku a opakováně provádíme dělení.
- **Výhody hierarchického shlukování:**
  - Možnost získat ucelený pohled pomocí dendrogramu a teprve na jeho základě vybírat vhodný počet shluků.
  - Hierarchická struktura — při zvýšení počtu shluků o jedničku dojde pouze k rozdělení některého ze stávajících, přičemž ostatní se nemění.
  - Kvalita vzniklého dendrogramu z pohledu zachování párových vzdáleností originálních dat může například být měřena pomocí **cophenetic correlation**.
- **Nevýhody — výpočetní náročnost:**
  - Hierarchické shlukování se příliš nehodí pro velké datové soubory, protože je výpočetně náročné.
  - V jednom kroku agglomerativního algoritmu je nutné provést  $m(m - 1)/2$  porovnání, kde  $m$  je aktuální počet shluků. Složitost je tedy  $\mathcal{O}(N^3)$ . V případě single nebo complete linkage je možné se dostat na  $\mathcal{O}(N^2)$ .
  - V jednom kroku divizního algoritmu je nutné provést  $2^{m-1}$  porovnání, kde  $m$  je velikost děleného shluku, což implikuje složitost  $\mathcal{O}(2^N)$ .

## 13 Shlukování pomocí algoritmu k-means

**Otázka ke zkoušce 15 (Shlukování pomocí algoritmu k-means):** Cíle nesupervizovaného učení a shlukování. Shlukování jako optimalizační úloha, algoritmus k-means a jeho účelová funkce. Vyhodnocování pomocí Silhouette skóre.

### 13.1 Nesupervizované učení a shluková analýza

- **Nesupervizované učení** (angl. unsupervised learning) nastává v situaci, kdy data nemáme nikterak označena. Tj. nemáme žádnou veličinu, kterou bychom u trénovacích dat znali a snažili se ji naučit predikovat. Cílem je porozumět struktuře dat pouze na základě nich samotných.
- **Shlukování** (angl. clustering) je jednou ze základních metod nesupervizovaného učení. Cílem je roztrídit data do skupin nazývaných **shluky** tak, že platí dva přirozené požadavky:
  - Blízké body budou ve stejném shluku.
  - Vzdálené body budou v různých shlucích.
- **Problém:** Tyto intuitivní požadavky obecně nejsou kompatibilní a neexistuje univerzální hodnotící kritérium kvality shlukování.

### 13.2 Shlukování jako optimalizační úloha

- Oblíbeným přístupem ke shlukování je formulace ve tvaru **optimalizační úlohy**.
- Je třeba definovat **účelovou funkci** (angl. objective function), která ohodnocuje daný rozklad množiny na shluky. Cílem je nalézt rozklad, který účelovou funkci minimalizuje.
- Pro dané  $k$  hledáme takový rozklad  $\mathcal{C} = (C_1, \dots, C_k)$  na prostoru  $\mathcal{X} = \mathbb{R}^p$  vybaveném Euklidovskou vzdáleností, který minimalizuje účelovou funkci:

$$G(\mathcal{C}) = \sum_{i=1}^k \frac{1}{2|C_i|} \sum_{\mathbf{x}, \mathbf{y} \in C_i} \|\mathbf{x} - \mathbf{y}\|^2$$

- V účelové funkci se pro každý shluk sečtou průměrné kvadráty vzdáleností všech bodů daného shluku od jeho ostatních bodů.

### 13.3 Souvislost účelové funkce s geometrickým středem

- **Tvrzení:** Pro konečnou množinu bodů  $A \subset \mathbb{R}^p$  platí:

$$\frac{1}{2|A|} \sum_{\mathbf{x}, \mathbf{y} \in A} \|\mathbf{x} - \mathbf{y}\|^2 = \sum_{\mathbf{x} \in A} \|\mathbf{x} - \bar{\mathbf{x}}\|^2 = \min_{\boldsymbol{\mu} \in \mathbb{R}^p} \sum_{\mathbf{x} \in A} \|\mathbf{x} - \boldsymbol{\mu}\|^2,$$

kde  $\bar{\mathbf{x}} = \frac{1}{|A|} \sum_{\mathbf{x} \in A} \mathbf{x}$  je **geometrický střed** (angl. centroid) množiny  $A$ .

- **Důkaz:** Pro každé  $\mathbf{a}, \mathbf{b} \in \mathbb{R}^p$  platí  $\|\mathbf{a} - \mathbf{b}\|^2 = (\mathbf{a} - \mathbf{b})^T(\mathbf{a} - \mathbf{b}) = \|\mathbf{a}\|^2 - 2\mathbf{a}^T\mathbf{b} + \|\mathbf{b}\|^2$ , protože  $\mathbf{b}^T\mathbf{a} = \mathbf{a}^T\mathbf{b}$ . Pro  $\mathbf{a} = \mathbf{x} - \boldsymbol{\mu}$  a  $\mathbf{b} = \mathbf{y} - \boldsymbol{\mu}$  z toho plyne:

$$\frac{1}{2|A|} \sum_{\mathbf{x}, \mathbf{y} \in A} \|\mathbf{x} - \mathbf{y}\|^2 = \frac{1}{2|A|} \sum_{\mathbf{x}, \mathbf{y} \in A} \|\mathbf{x} - \boldsymbol{\mu}\|^2 + \frac{1}{2|A|} \sum_{\mathbf{x}, \mathbf{y} \in A} \|\mathbf{y} - \boldsymbol{\mu}\|^2 - \frac{1}{|A|} \sum_{\mathbf{x}, \mathbf{y} \in A} (\mathbf{x} - \boldsymbol{\mu})^T(\mathbf{y} - \boldsymbol{\mu}).$$

Poslední člen upravíme:

$$\frac{1}{|A|} \sum_{\mathbf{x}, \mathbf{y} \in A} (\mathbf{x} - \boldsymbol{\mu})^T(\mathbf{y} - \boldsymbol{\mu}) = \frac{1}{|A|} \sum_{\mathbf{x} \in A} (\mathbf{x} - \boldsymbol{\mu})^T \sum_{\mathbf{y} \in A} (\mathbf{y} - \boldsymbol{\mu}) = \frac{1}{|A|} \left\| \sum_{\mathbf{x} \in A} (\mathbf{x} - \boldsymbol{\mu}) \right\|^2.$$

Poslední člen je tedy vždy nezáporný. Navíc je rovný nule právě, když  $\boldsymbol{\mu} = \bar{\mathbf{x}}$ , což plyne z definice  $\bar{\mathbf{x}}$ . Prohodíme-li ve druhém členu  $\mathbf{x}$  a  $\mathbf{y}$ , dostaneme první člen, a po vysčítání přes  $\mathbf{y}$  dostaneme:

$$\frac{1}{2|A|} \sum_{\mathbf{x}, \mathbf{y} \in A} \|\mathbf{x} - \mathbf{y}\|^2 \leq \sum_{\mathbf{x} \in A} \|\mathbf{x} - \boldsymbol{\mu}\|^2, \quad \text{s rovností pouze pokud } \boldsymbol{\mu} = \bar{\mathbf{x}}. \quad \square$$

- Na základě tohoto tvrzení můžeme účelovou funkci vyjádřit jako:

$$G(\mathcal{C}) = \sum_{i=1}^k \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \bar{\mathbf{x}}_i\|^2,$$

kde  $\bar{\mathbf{x}}_i$  je geometrický střed  $i$ -tého shluku.

## 13.4 Algoritmus k-means

- **Poznámka (NP-těžkost):** Problém nalezení globálního minima účelové funkce je výpočetně obtížný — je **NP-těžký** [Aloise et al. (2009)]. NP-těžký problém je takový, pro který neexistuje známý polynomiální algoritmus a předpokládá se, že ani existovat nemůže. V praxi to znamená, že pro velké instance nelze nalézt optimální řešení v rozumném čase.
- **Algoritmus k-means** je heuristický iterativní algoritmus, který v každém kroku zmenšuje hodnotu účelové funkce a konverguje tak k jejímu **lokálnímu minimu**.
- **Slovní popis algoritmu:**
  1. Nejprve zvolíme  $k$  středových bodů.
  2. Iterativně opakujeme:
    - (i) Vytvoříme shluky odpovídající středovým bodům tak, že pro každý bod  $\mathbf{x}$  najdeme k němu nejbližší středový bod a podle něj  $\mathbf{x}$  zařadíme do shluku.
    - (ii) Spočítáme nové středové body jako geometrické středy těchto shluků.
- **Formalizace algoritmu k-means:**
  - **Inicializace:** Počáteční rozmístění  $k$  středových bodů  $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_k$ .
  - **Iterativní část:**
    - (i) Roztrídíme body do shluků:  $C_i = \{\mathbf{x} \in \mathcal{D} \mid i = \arg \min_j \|\mathbf{x} - \boldsymbol{\mu}_j\|\}$ .
    - (ii) Přepočítáme body  $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_k$  jako geometrické středy těchto shluků:  $\boldsymbol{\mu}_i \leftarrow \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \mathbf{x}$ .
- Běh algoritmu zastavíme, jakmile je změna hodnoty účelové funkce mezi jednotlivými iteracemi dostatečně malá.
- Výsledek algoritmu významně závisí na **inicializační části**. Počáteční středové body jsou obvykle generovány náhodně (např. náhodným výběrem z dat). Existují i “chytrější metody” jako např. **k-means++**.
- Algoritmus je následně opakován spouštěn. Jako finální pak bereme výsledek běhu s nejnižší hodnotou účelové funkce.

## 13.5 Důkaz lokální optimalizace

- Zafixujme  $\boldsymbol{\mu}_i = \bar{\mathbf{x}}_i$ . Vytvořme nové shluky  $\tilde{\mathcal{C}} = \{\tilde{C}_1, \dots, \tilde{C}_k\}$  tak, že bod  $\mathbf{x}$  přesuneme do takového shluku  $\tilde{C}_i$ , ve kterém je vzdálenost  $\|\mathbf{x} - \boldsymbol{\mu}_i\|$  nejmenší.
- Tím zcela jistě dojde ke zmenšení součtu kvadrátů vzdáleností:

$$\sum_{i=1}^k \sum_{\mathbf{x} \in \tilde{C}_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 \leq \sum_{i=1}^k \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2.$$

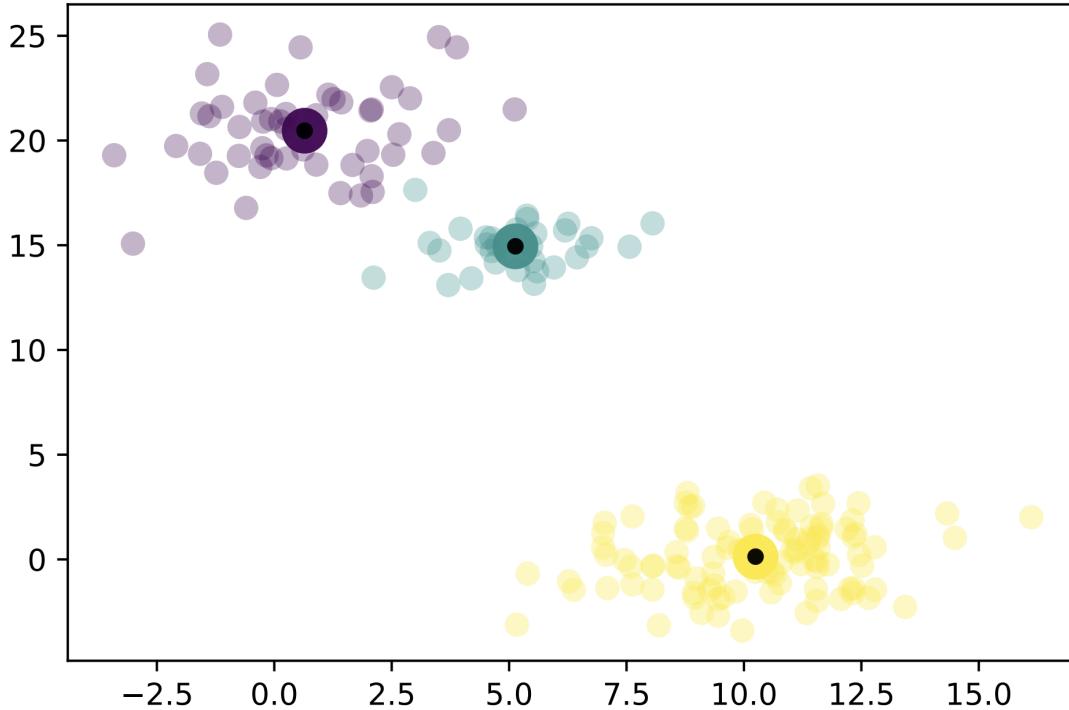
- Z druhé rovnosti předchozího tvrzení pak plyne:

$$G(\tilde{\mathcal{C}}) = \sum_{i=1}^k \sum_{\mathbf{x} \in \tilde{C}_i} \|\mathbf{x} - \bar{\mathbf{x}}_i\|^2 \leq \sum_{i=1}^k \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2,$$

kde  $\bar{\mathbf{x}}_i$  je geometrický střed shluku  $\tilde{C}_i$ .

- Dohromady tedy máme  $G(\tilde{\mathcal{C}}) \leq G(\mathcal{C})$ . Tento postup můžeme opakovat a postupně tak klesat k nějakému lokálnímu minimu účelové funkce.

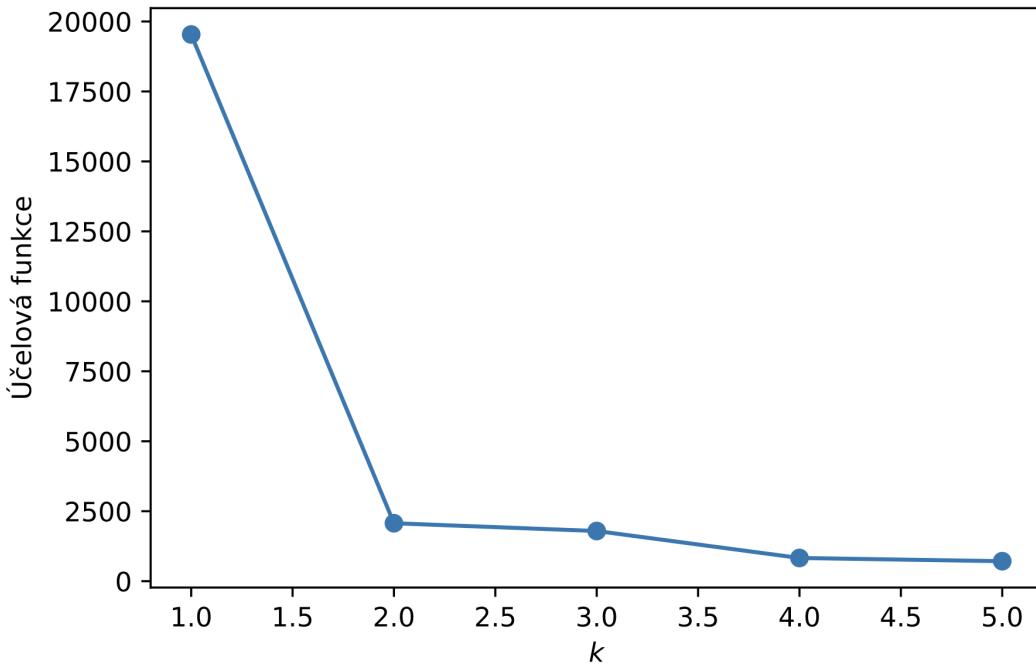
### 13.6 Ukázka výsledku algoritmu k-means



Obrázek 23: Výsledné shlukování včetně geometrických středů shluků

### 13.7 Problematika volby $k$

- Na rozdíl od hierarchického shlukování, kde lze počet shluků určovat až na základě dendrogramu, je u algoritmu k-means nezbytné stanovit  $k$  **dopředu**.
- Bohužel neexistuje žádný univerzální způsob, jak počet shluků určit automaticky.
- **Metoda lokte** (angl. elbow method):
  - Je-li  $k^*$  optimální počet shluků, lze očekávat, že pro  $k < k^*$  bude účelová funkce s měnícím se  $k$  klesat hodně. Pro  $k \geq k^*$  lze naopak očekávat zmenšení poklesu účelové funkce.
  - Optimální  $k$  tedy můžeme detektovat jako hodnotu, pro kterou se mění pokles účelové funkce z hodně prudkého na méně prudký — hledat tzv. **loket**.
  - Je to ale hodně subjektivní a pro některá data v podstatě nepoužitelné.



Obrázek 24: Metoda lokte — závislost účelové funkce na  $k$

Na obrázku 24 bychom nejspíš určili jako optimální  $k = 2$ . Přitom ve skutečnosti byla data vygenerována jako směs tří různých dvouozměrných normálních rozdělení.

### 13.8 Vyhodnocování pomocí Silhouette skóre

- **Silhouette** [Rousseeuw (1987)] je metoda pro evaluaci shlukování, kterou lze využít i k určení optimálního počtu shluků.
- Uvažujme shlukování  $\mathcal{D} = C_1 \cup \dots \cup C_k$  na metrickém prostoru  $\mathcal{X}$  s metrikou  $d(\mathbf{x}, \mathbf{y})$  a pro libovolný bod  $\mathbf{x} \in \mathcal{D}$  označme  $j(\mathbf{x})$  index shluku, do kterého  $\mathbf{x}$  patří.
- **Výpočet Silhouette skóre pro bod  $\mathbf{x}$ :**

- **Vnitřní rozdílnost** (angl. within dissimilarity) — průměrná vzdálenost bodu  $\mathbf{x}$  od všech ostatních bodů ve stejném shluku:

$$a(\mathbf{x}) = \frac{1}{|C_{j(\mathbf{x})}| - 1} \sum_{\mathbf{y} \in C_{j(\mathbf{x})}, \mathbf{y} \neq \mathbf{x}} d(\mathbf{x}, \mathbf{y})$$

- Pro každý další shluk  $C_i$ ,  $i \neq j(\mathbf{x})$  spočteme průměrnou vzdálenost:

$$d(\mathbf{x}, C_i) = \frac{1}{|C_i|} \sum_{\mathbf{y} \in C_i} d(\mathbf{x}, \mathbf{y})$$

- **Sousední rozdílnost** (angl. between dissimilarity) — minimum z průměrných vzdáleností od ostatních shluků:

$$b(\mathbf{x}) = \min_{i \neq j(\mathbf{x})} d(\mathbf{x}, C_i)$$

- **Silhouette skóre bodu  $\mathbf{x}$ :**

$$s(\mathbf{x}) = \frac{b(\mathbf{x}) - a(\mathbf{x})}{\max\{a(\mathbf{x}), b(\mathbf{x})\}}$$

Jestliže máme pouze jeden shluk, položíme  $s(\mathbf{x}) = 0$ .

- **Interpretace:** Vždy platí  $-1 \leq s(\mathbf{x}) \leq 1$ .

- $s(\mathbf{x}) \approx 1$ : Vnitřní rozdílnost  $a(\mathbf{x})$  je mnohem menší než sousední rozdílnost  $b(\mathbf{x})$  — bod je dobře zatříděn.
- $s(\mathbf{x}) \approx 0$ :  $a(\mathbf{x})$  je podobně velké jako  $b(\mathbf{x})$  — bod je na okraji svého shluku.
- $s(\mathbf{x}) \approx -1$ :  $a(\mathbf{x})$  je mnohem větší než  $b(\mathbf{x})$  — bod je špatně přiřazen a měl by patřit do sousedního shluku.

- Průměrné skóre pro shluk  $C_i$ :

$$s_i = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} s(\mathbf{x})$$

- Průměrné skóre pro celé shlukování:

$$s = \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} s(\mathbf{x})$$

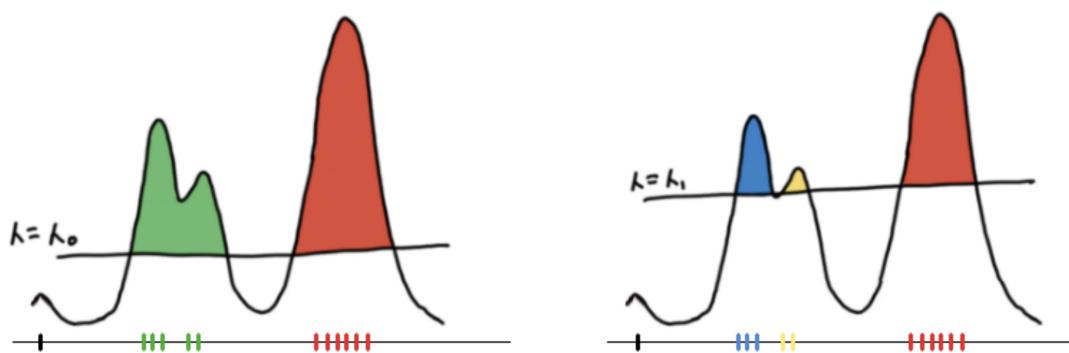
- Čím vyšší hodnota  $s$ , tím je celé shlukování lepší. Porovnání skóre pro různé počty shluků lze použít k nalezení optimálního  $k$  jako hodnoty, při které je skóre  $s$  **maximální**.

## 14 Shlukování pomocí algoritmu DBSCAN

**Otzávka ke zkoušce 16 (Shlukování pomocí algoritmu DBSCAN):** Cíle nesupervizovaného učení a shlukování. Algoritmus DBSCAN.

### 14.1 Úvod do shlukování pomocí hustoty

- V této sekci se zaměříme na shlukování, které využívá odhad hustoty rozdělení dat na prostoru  $\mathcal{X}$  možných hodnot příznaků.
- Z pravděpodobnostního pohledu chápeme pozorovaná data jako realizace náhodného vektoru  $\mathbf{X} = (X_1, \dots, X_p)^T$ .
- Pokud budeme mít nějakým způsobem odhadnutou hustotu pravděpodobnosti  $f_X(\mathbf{x}) \equiv f_X(x_1, \dots, x_p)$  (resp. něco, co jí je úměrné), můžeme ji využít k získání shlukování včetně identifikace bodů, které do žádných shluků nepatří.
- Shluky můžeme získat jako souvislé oblasti, ve kterých odhad hustoty překročí nějakou zvolenou hranici.
- Body mimo tyto oblasti pak označíme jako **šum** (angl. noise).

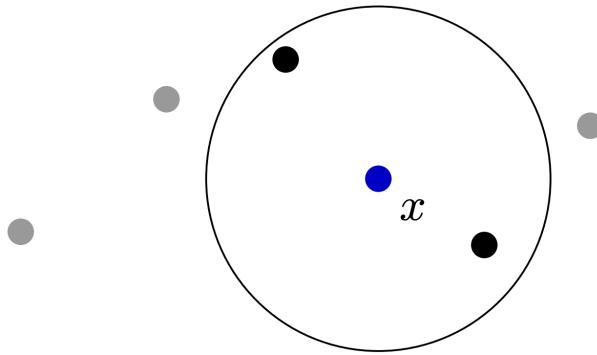


Obrázek 25: Shlukování pomocí hustoty — shluky jako oblasti s vysokou hustotou

## 14.2 DBSCAN: úvodní pojmy

- Jedním z aktuálně nejpoužívanějších algoritmů shlukování, který je implicitně založen na principu odhadu hustoty, je algoritmus **DBSCAN**, což je zkratka angl. **density-based spatial clustering of applications with noise**, [Ester et al. (1996)].
- Připravme si nyní základní pojmy, na kterých DBSCAN stojí.
- Uvažujme metrický prostor  $\mathcal{X}$  s metrikou  $d(\mathbf{x}, \mathbf{y})$  ze kterého pochází dataset  $\mathcal{D}$  a parametry  $\varepsilon > 0$  a  $\text{MinPts} \in \mathbb{N}^+$ .
- Definujme  $\varepsilon$  **okolí** bodu  $\mathbf{x}$  v  $\mathcal{D}$  (angl.  $\varepsilon$ -neighborhood) jako množinu:

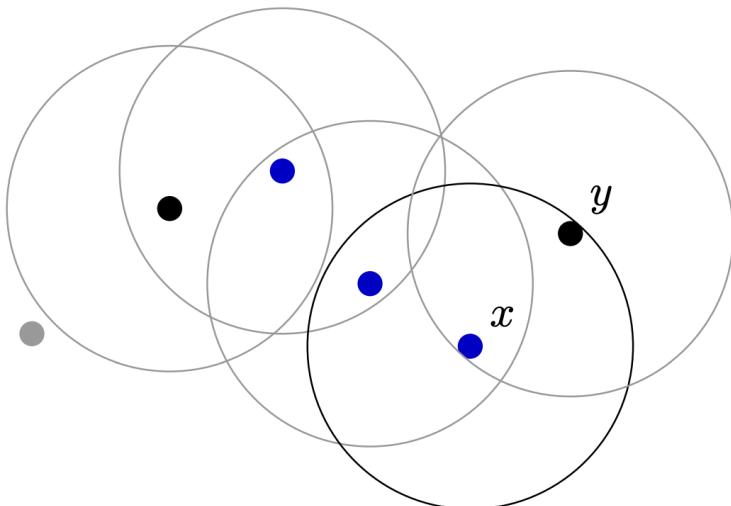
$$N_\varepsilon(\mathbf{x}) = \{\mathbf{y} \in \mathcal{D} \mid d(\mathbf{x}, \mathbf{y}) \leq \varepsilon\}$$



Obrázek 26: Znázornění  $\varepsilon$  okolí

## 14.3 DBSCAN: klíčové body, přímá dosažitelnost

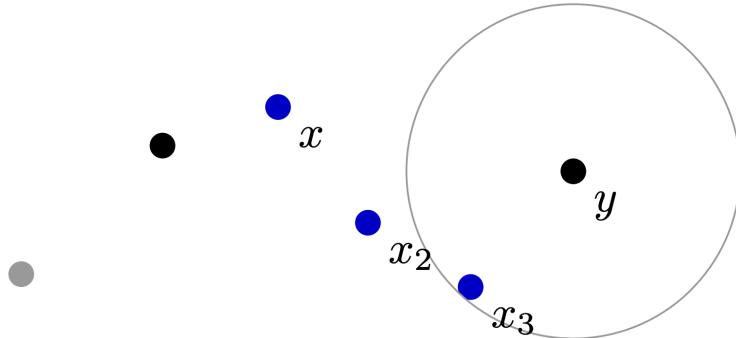
- Bod  $\mathbf{x} \in \mathcal{D}$  je **klíčový bod** (angl. core point), jestliže v jeho  $\varepsilon$  okolí v  $\mathcal{D}$  je alespoň  $\text{MinPts}$  bodů:
$$|N_\varepsilon(\mathbf{x})| \geq \text{MinPts}$$
- Bod  $\mathbf{y} \in \mathcal{D}$  je **přímo dosažitelný** (angl. directly density-reachable) z bodu  $\mathbf{x} \in \mathcal{D}$ , jestliže  $\mathbf{x}$  je klíčový bod a  $\mathbf{y} \in N_\varepsilon(\mathbf{x})$ .
- Relace přímé dosažitelnosti je symetrická pro dvojici klíčových bodů, je ale nesymetrická pro tzv. **okrajový bod** (angl. border point), což je bod, který není klíčový, ale je přímo dosažitelný z klíčového bodu.



Obrázek 27: Pro  $\text{MinPts} = 3$  je bod  $y$  přímo dosažitelný z  $x$ . Všechny klíčové body jsou modré, všechny okrajové body jsou černé.

## 14.4 DBSCAN: dosažitelnost

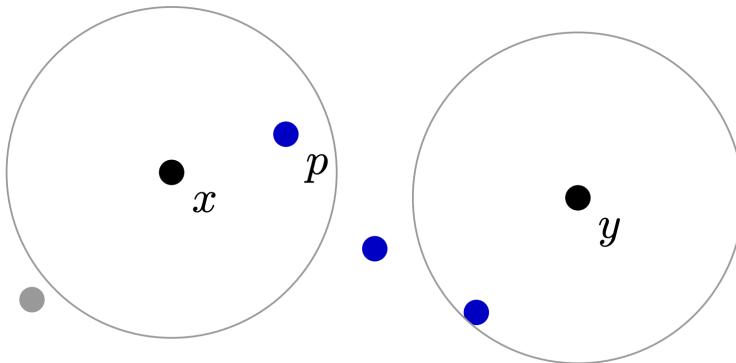
- Bod  $y \in \mathcal{D}$  je **dosažitelný** (angl. density-reachable) z bodu  $x \in \mathcal{D}$ , pokud existuje posloupnost  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n \in \mathcal{D}$  bodů tak, že  $\mathbf{x}_1 = x$ ,  $\mathbf{x}_n = y$  a pro každé  $i = 1, \dots, n - 1$  je  $\mathbf{x}_{i+1}$  přímo dosažitelný z bodu  $\mathbf{x}_i$ .
- Z toho plyne, že všechny body po cestě kromě posledního musí být klíčové body.



Obrázek 28: Pro  $\text{MinPts} = 3$  je bod  $y$  dosažitelný z bodu  $x$

## 14.5 DBSCAN: spojenost

- Bod  $y \in \mathcal{D}$  je **spojený** (angl. density-connected) s bodem  $x \in \mathcal{D}$ , jestliže existuje (klíčový bod)  $p \in \mathcal{D}$  tak, že  $x$  i  $y$  jsou dosažitelné z bodu  $p$ .
- Relace spojenosti je zjevně symetrická. Pro klíčové body je také tranzitivní.
- Jestliže jsou dva body spojené a jeden z nich je klíčový bod, tak ten druhý je z toho prvního dosažitelný.

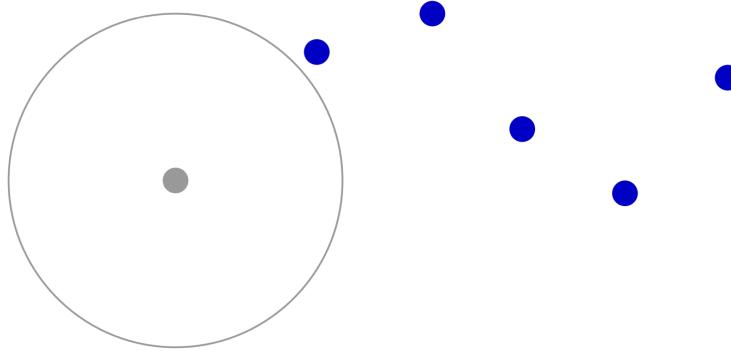


Obrázek 29: Pro  $\text{MinPts} = 3$  je bod  $y$  spojený s bodem  $x$

## 14.6 DBSCAN: shluky a šum

- Shluk nyní definujeme jako maximální množinu spojených bodů.
- **Definice:** **Shluk** (angl. cluster)  $C$  je podmnožina  $\mathcal{D}$  obsahující alespoň jeden klíčový bod tak, že:
  - Pro každé  $\mathbf{x}, \mathbf{y} \in C$  platí, že když  $\mathbf{x} \in C$  a  $\mathbf{y}$  je dosažitelný z  $\mathbf{x}$ , pak  $\mathbf{y} \in C$  (**maximalita**).
  - Pro každé  $\mathbf{x}, \mathbf{y} \in C$  je  $\mathbf{x}$  spojený s  $\mathbf{y}$  (**souvislost**).
- Označme  $C_1, \dots, C_k$  množinu všech shluků v  $\mathcal{D}$  (vzhledem k  $\varepsilon$  a  $\text{MinPts}$ ). Množinu  $N$  bodů z  $\mathcal{D}$ , které nejsou v žádném ze shluků nazýváme **šumem** (angl. noise):

$$N = \mathcal{D} \setminus \bigcup_{i=1}^k C_i$$



Obrázek 30: Body ve shluku jsou modré, bod šumu je šedivý

### 14.7 Poznámky k definici shluků

- Shluk  $C$  vždy obsahuje nějaký klíčový bod  $\mathbf{p} \in C$ , ze kterého jsou dosažitelné všechny body v jeho  $\varepsilon$  okolí, kterých je alespoň MinPts. Právě jsme tedy ukázali, že **každý shluk obsahuje alespoň MinPts bodů**.
- Každý bod ve shluku  $C$  je spojený se všemi klíčovými body v  $C$  a tedy je z libovolného klíčového bodu dosažitelný. Shluk je tedy tvořen všemi body, které jsou dosažitelné z libovolného klíčového bodu v  $C$ .
- To nám dává návod, jak může algoritmus tvorby shluků fungovat. Najdeme klíčový bod a vytvoříme k němu shluk jako množinu všech z něho dosažitelných bodů (které ještě nejsou v jiném shluku).

### 14.8 Abstraktní popis algoritmu DBSCAN

- **Algoritmus DBSCAN:**

1. **Nalezení klíčových bodů:** Spočítáme  $\varepsilon$  okolí každého bodu a identifikujeme klíčové body.
2. **Vytvoření zárodků shluků:** Spojíme sousední (přímo dosažitelné) klíčové body do shluků.
3. **Pro každý bod, který není klíčový:**
  - Přidáme do shluku podle klíčového bodu v jeho okolí.
  - Pokud takový neexistuje, přidáme mezi šum.

- **Poznámky k algoritmu:**

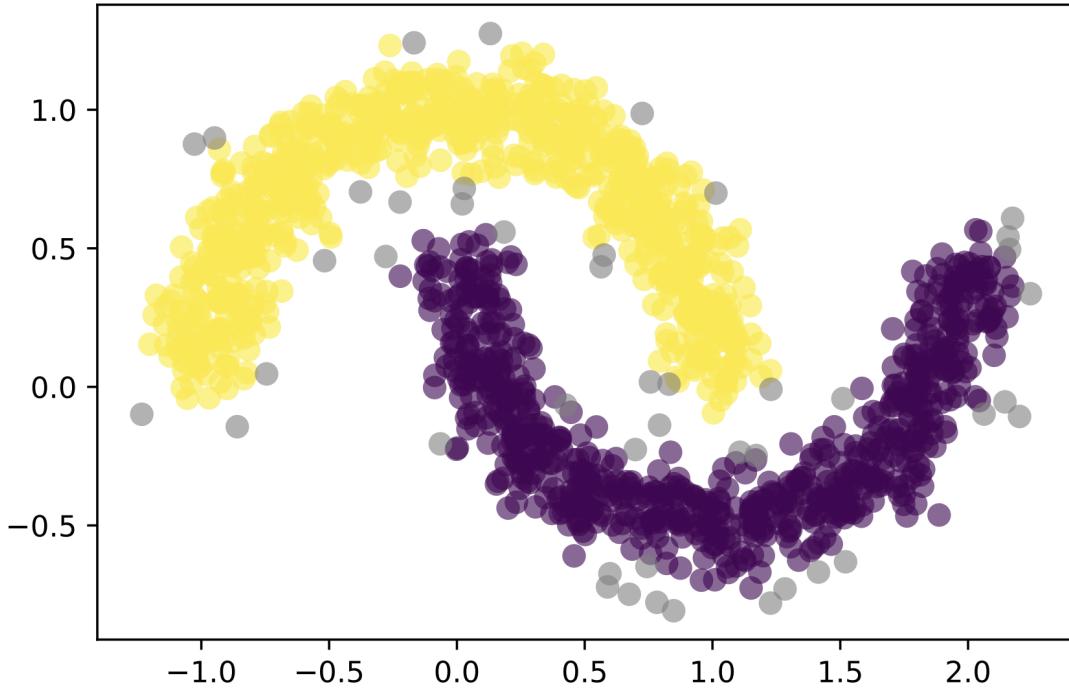
- Okrajový bod, který má ve svém  $\varepsilon$  okolí klíčové body z různých (zárodků) shluků spadne do prvního z těchto shluků, ke kterému se algoritmus dostane.
- Finálním výsledkem v takovém případě budou shluky, které nesplňují podmínu maximality v předchozí definici, protože okrajový bod bude pouze v jednom shluku a nikoliv ve všech, kde by dle maximality měl být.
- Lze ukázat ([Schubert et al. (2017)]), že složitost v nejhorším případě je  $\mathcal{O}(n^2)$ . V mnoha reálných situacích se ale lze dostat na  $\mathcal{O}(n \log n)$ .

### 14.9 DBSCAN: volba parametrů

- Zaměřme se nyní krátce na volbu parametrů algoritmu DBSCAN.
- Ukazuje se, že parametr MinPts je mnohem méně důležitý než parametr  $\varepsilon$ . Obvykle dobrou volbou jsou hodnoty okolo 4 – 6 (někdy bývá doporučováno  $2 \cdot p$ , kde  $p$  je počet příznaků).
- Pro parametr  $\varepsilon$  bývá doporučováno volit co nejmenší hodnotu s tím, že lze například brát průměrnou vzdálenost bodů k jejich nejbližšímu  $(2 \cdot p - 1)$ -tému sousedovi.
- Také lze sledovat velikost šumu. Uvádí se, že obvykle by poměr šumu měl být mezi 1% a 30%.
- Dále je dobré sledovat velikost největšího shluku. Pokud jeho velikost překračuje 50% velikosti datasetu, bývá vhodné zmenšit  $\varepsilon$ , případně použít nějaký pokročilejší algoritmus (např. HDBSCAN).
- Více detailů k volbě parametrů najdete v [Schubert et al. (2017)] nebo v [Sander et al. (1998)].

## 14.10 DBSCAN: ukázka a poznámky

- Mezi hlavní **výhody** DBSCAN (a obecně metod založených na hustotě) patří možnost nalezení **nekonvexních shluků**.
- Další výhodou je snížená citlivost na odlehlé hodnoty — které jsou označeny jako šum.



Obrázek 31: Znázornění shluků a šumu — DBSCAN dokáže nalézt nekonvexní shluky

## 14.11 Evaluace pomocí Silhouette skóre

- **Silhouette** [Rousseeuw, P. (1987)] je metoda pro evaluaci shlukování, kterou lze využít i k určení optimálního počtu shluků.
- Uvažujme shlukování  $\mathcal{D} = C_1 \cup \dots \cup C_k$  na metrickém prostoru  $\mathcal{X}$  s metrikou  $d(\mathbf{x}, \mathbf{y})$  a pro libovolný bod  $\mathbf{x} \in \mathcal{D}$  označme  $j(\mathbf{x})$  index shluku, do kterého  $\mathbf{x}$  patří, tj.  $\mathbf{x} \in C_{j(\mathbf{x})}$ .
- **Výpočet Silhouette skóre pro bod  $\mathbf{x}$ :**

- **Vnitřní rozdílnost** (angl. within dissimilarity) — průměrná vzdálenost bodu  $\mathbf{x}$  od všech ostatních bodů ve stejném shluku:

$$a(\mathbf{x}) = \frac{1}{|C_{j(\mathbf{x})}| - 1} \sum_{\mathbf{y} \in C_{j(\mathbf{x})}, \mathbf{y} \neq \mathbf{x}} d(\mathbf{x}, \mathbf{y})$$

- Pro každý další shluk  $C_i$ ,  $i \neq j(\mathbf{x})$  spočteme průměrnou vzdálenost:

$$d(\mathbf{x}, C_i) = \frac{1}{|C_i|} \sum_{\mathbf{y} \in C_i} d(\mathbf{x}, \mathbf{y})$$

- **Sousední rozdílnost** (angl. between dissimilarity) — minimum z průměrných vzdáleností od ostatních shluků:

$$b(\mathbf{x}) = \min_{i \neq j(\mathbf{x})} d(\mathbf{x}, C_i)$$

- **Silhouette skóre bodu  $\mathbf{x}$ :**

$$s(\mathbf{x}) = \frac{b(\mathbf{x}) - a(\mathbf{x})}{\max\{a(\mathbf{x}), b(\mathbf{x})\}}$$

Jestliže máme pouze jeden shluk, položíme  $s(\mathbf{x}) = 0$ .

- **Interpretace:** Vždy platí  $-1 \leq s(\mathbf{x}) \leq 1$ .

- $s(\mathbf{x}) \approx 1$ : Vnitřní rozdílnost  $a(\mathbf{x})$  je mnohem menší než sousední rozdílnost  $b(\mathbf{x})$  — bod je **dobře zatříden**.
- $s(\mathbf{x}) \approx 0$ :  $a(\mathbf{x})$  je podobně velké jako  $b(\mathbf{x})$  — bod je **na okraji svého shluku** a sousední shluk je blízko.
- $s(\mathbf{x}) \approx -1$ :  $a(\mathbf{x})$  je mnohem větší než  $b(\mathbf{x})$  — bod je **špatně přiřazen** a měl by spíše patřit do sousedního shluku.

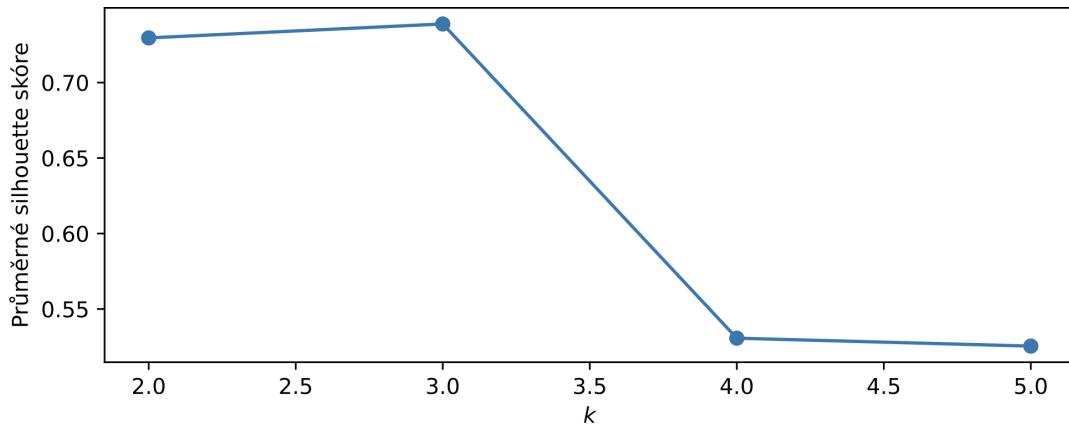
- **Průměrné skóre pro shluk  $C_i$ :**

$$s_i = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} s(\mathbf{x})$$

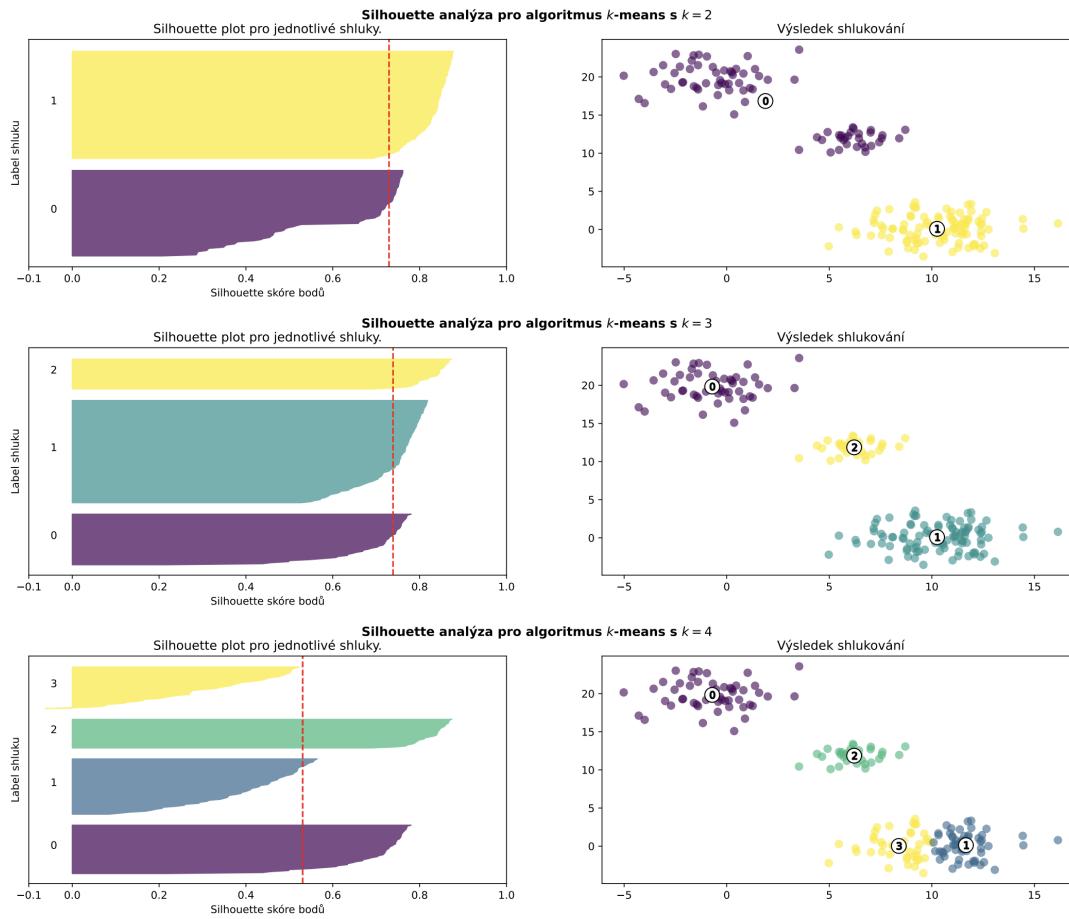
- **Průměrné skóre pro celé shlukování:**

$$s = \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} s(\mathbf{x})$$

- Čím vyšší hodnotu  $s$  dostaneme, tím jsou body ve shlucích správněji umístěny a tedy je celé shlukování lepší.
- Porovnání skóre pro různé počty shluků lze použít k nalezení optimálního  $k$  jako hodnoty, při které je skóre  $s$  **maximální**.



Obrázek 32: Závislost průměrného silhouettko skóre na počtu shluků  $k$



Obrázek 33: Silhouette analýza pro algoritmus k-means s  $k = 2, 3, 4$

## 15 Nesupervizované učení, asociační pravidla

**Otázka ke zkoušce 17 (Nesupervizované učení, asociační pravidla):** Cíle nesupervizovaného učení. Asociační pravidla.

### 15.1 Nesupervizované učení (Unsupervised Learning)

- **Nesupervizované učení** (angl. unsupervised learning) nastává v situaci, kdy data nemáme nikterak označena. Tj. nemáme žádnou veličinu, kterou bychom u trénovacích dat znali a snažili se ji naučit predikovat.
- Cílem nesupervizovaného učení je porozumět struktuře dat pouze na základě nich samotných. To znamená bez nějakého vnějšího vodítka. Proto se nesupervizovanému učení také říká **učení bez učitele**.
- Porozuměním zde typicky rozumíme nalezení co „nejmenších“ oblastí v prostoru příznaků, kde se data vyskytují nejčastěji.
- Obvykle totiž platí, že se naměřená skutečná data nevyskytují v celém prostoru stejně pravděpodobně, ale bývají více či méně lokalizována — tvoří nějaké shluky, vyskytují se v méně-dimenzionálních oblastech atd.
- Porozumění této lokalizaci přináší důležitou informaci o vnitřní struktuře dat!
- **Obecný problém nesupervizovaného učení:** Vůbec není jasné, jak bychom měli vyhodnocovat úspěšnost získaného porozumění. To je velký rozdíl oproti supervizovanému učení, kde je možné kvalitu naučeného modelu vyhodnocovat mnoha víceméně rovnocennými způsoby (např. přesností u klasifikace).

## 15.2 Analýza asociačních pravidel

- **Analýza asociačních pravidel** je jedna ze standardních a oblíbených metod pro dobývání znalostí z komerčních databází.
- Obecným cílem je nalézt společné hodnoty příznaků  $\mathbf{X} = (X_1, \dots, X_p)^T$ , které se v databázi nejčastěji vyskytují.
- V zásadě jde přesně o jeden z hlavních cílů nesupervizovaného učení, kdy chceme nalézt oblasti prostoru, kde se data vyskytují s velkou pravděpodobností.
- V nejčastěji používaném zjednodušení se zabýváme pouze oblastmi, které jsou ve tvaru kartézského součinu pro jednotlivé příznaky, tj. chceme, aby pravděpodobnost:

$$P\left(\bigcap_{j=1}^p (X_j \in s_j)\right),$$

kde  $s_j$  je podmnožina hodnot příznaku  $X_j$ , byla relativně velká.

- Průnik  $\bigcap_{j=1}^p (X_j \in s_j)$  se v takovém případě nazývá **konjunktivní pravidlo** (angl. conjunctive rule).

## 15.3 Analýza nákupního košíku

- **Analýza nákupního košíku** (angl. market basket analysis) je nejčastěji aplikována v případě binárních příznaků,  $X_j \in \{0, 1\}$ .
- Pro oblast, kterou hledáme ve tvaru kartézského součinu se v tomto případě používá ještě další omezení, a to, že  $s_j$  je buď jednoprvková množina  $\{1\}$  nebo všechny možnosti daného příznaku  $X_j$  (v tu chvíli příznak vypadne).
- Ekvivalentně tedy hledáme množinu indexů  $\mathcal{K} \subset \{1, \dots, p\}$  tak, že:

$$P\left(\bigcap_{j \in \mathcal{K}} (X_j = 1)\right) = P\left(\prod_{j \in \mathcal{K}} X_j = 1\right)$$

je relativně velká.

- Množina  $\mathcal{K}$  se pak nazývá **množina položek** (angl. item set).
- Relativní velikost položek v datasetu, které danou množinu položek obsahují se značí  $T(\mathcal{K})$  a nazývá **podpora** (angl. support) množiny položek  $\mathcal{K}$  a odpovídá odhadu výše uvedené pravděpodobnosti:

$$T(\mathcal{K}) = \hat{P}\left(\bigcap_{j \in \mathcal{K}} (X_j = 1)\right) = \frac{1}{N} \sum_{i=1}^N \prod_{j \in \mathcal{K}} x_{i;j}$$

## 15.4 Asociační pravidla: definice

- Při provádění analýzy nákupního košíku hledáme všechny množiny položek, pro které je podpora větší než nějaká zvolená mez  $t$ :

$$\{\mathcal{K}_\ell \mid T(\mathcal{K}_\ell) > t\}$$

- K nalezení řešení se používá efektivní algoritmus (případně jeho novější vylepšení), který se nazývá **Apriori algoritmus** [Agrawal, Srikant (1994)].
- Pro každou množinu položek  $\mathcal{K}$ , kterou takto získáme, dále hledáme vhodné rozložení na dvě disjunktní podmnožiny  $A$  a  $B$ ,  $A \cup B = \mathcal{K}$ , které budeme nazývat **asociační pravidlo** (angl. association rule) a značit:

$$A \Rightarrow B$$

- První položka  $A$  asociačního pravidla se nazývá **předpoklad** (angl. antecedent) a druhá položka  $B$  se nazývá **závěr** (angl. consequent).

- **Podpora**  $T(A \Rightarrow B)$  pravidla  $A \Rightarrow B$  je definována jako podpora sjednocení  $\mathcal{K} = A \cup B$ .
- **Spolehlivost** (angl. confidence)  $C(A \Rightarrow B)$  pravidla  $A \Rightarrow B$  je definována jako podpora pravidla podělená podporou předpokladu  $A$ :

$$C(A \Rightarrow B) = \frac{T(A \Rightarrow B)}{T(A)},$$

což odpovídá odhadu podmíněné pravděpodobnosti  $P(B | A)$ .

### 15.5 Asociační pravidla: další míry

- Asociační pravidla jsou volena tak, aby spolehlivost byla větší než nějaká zvolená mez  $c$ .
- Finálním výstupem asociační analýzy pravidel je množina asociačních pravidel, které splňují:

$$T(A \Rightarrow B) > t \quad \text{a} \quad C(A \Rightarrow B) > c$$

- U nalezených asociačních pravidel se dále často měří **zdvih** (angl. lift) definovaný jako:

$$L(A \Rightarrow B) = \frac{C(A \Rightarrow B)}{T(B)},$$

který odpovídá odhadu podílu  $\frac{P(B|A)}{P(B)}$ , což znamená, kolikanásobně je větší  $P(B | A)$  oproti  $P(B)$ .

- Někdy se také měří **pokrytí** (angl. coverage) definované jako:

$$\text{Coverage}(A \Rightarrow B) = \frac{T(A \Rightarrow B)}{T(B)},$$

což odpovídá odhadu podmíněné pravděpodobnosti  $P(A | B)$ .

- Pokrytí tedy indikuje, jak často lze závěr vyložit coby důsledek předpokladu.

### 15.6 Asociační pravidla: příklady

- Klasickým příkladem asociačního pravidla je:

$$\{\text{párky}\} \Rightarrow \{\text{hořčice, chléb}\}$$

- **Podpora** 0.06 znamená, že v celém datasetu se trojice položek  $\{\text{párky, hořčice, chléb}\}$  vyskytuje v 6% případů.
- Pokud se páry vyskytují v datasetu v 8% případů, bude **spolehlivost**  $0.06/0.08 = 0.75$ , což znamená, že když si zákazník koupil páry, v 75% případů si koupil také hořčici a chléb.
- Pokud je dvojice hořčice a chléb v datasetu v 15% případů, **zdvih** bude  $0.75/0.15 = 5$ .
- **Pokrytí** v takovém případě bude  $0.06/0.15 = 0.4$ . Čili ve 40% případů lze hořčici a chléb uvažovat jako důsledek koupě pároků.
- **Nevýhoda omezení na podporu:** Pravidla s velkou hodnotou spolehlivosti i zdvihu ale s nízkou podporou, jako např.:

$$\{\text{doutník}\} \Rightarrow \{\text{rum}\},$$

nebudou nalezeny.

## 15.7 Shrnutí asociačních pravidel

- **Podpora** (angl. support): Jak často se množina položek vyskytuje v datech.

$$T(\mathcal{K}) = \frac{\text{počet transakcí obsahujících } \mathcal{K}}{\text{celkový počet transakcí}}$$

- **Spolehlivost** (angl. confidence): Jak často se závěr vyskytuje, pokud se vyskytuje předpoklad.

$$C(A \Rightarrow B) = \frac{T(A \cup B)}{T(A)} = \hat{P}(B | A)$$

- **Zdvih** (angl. lift): Kolikanásobně je pravděpodobnější závěr při splnění předpokladu oproti samostatnému výskytu závěru.

$$L(A \Rightarrow B) = \frac{C(A \Rightarrow B)}{T(B)} = \frac{P(B | A)}{P(B)}$$

- **Pokrytí** (angl. coverage): Jak často lze závěr vyložit jako důsledek předpokladu.

$$\text{Coverage}(A \Rightarrow B) = \frac{T(A \cup B)}{T(B)} = \hat{P}(A | B)$$

- **Interpretace zdvihu:**

- $L > 1$ : Předpoklad a závěr jsou pozitivně korelované.
- $L = 1$ : Předpoklad a závěr jsou nezávislé.
- $L < 1$ : Předpoklad a závěr jsou negativně korelované.

## 16 Zajímavá videa na YouTube přesahující rámec kurzu

### 16.1 Vybraná videa z pravděpodobnosti a statistiky

- But what is the Central Limit Theorem? (3Blue1Brown) [31 min]
- Bayes theorem, the geometry of changing beliefs (3Blue1Brown) (část I) [15 min]
- The medical test paradox, and redesigning Bayes' rule (3Blue1Brown) (část II) [21 min]
- Why  $\pi$  is in the normal distribution (beyond integral tricks) (3Blue1Brown) [24 min]
- But what is a convolution? (3Blue1Brown) [23 min]
- Convolutions | Why X+Y in probability is a beautiful mess (3Blue1Brown) [27 min]
- A pretty reason why Gaussian + Gaussian = Gaussian (3Blue1Brown) [13 min]
- Binomial distributions | Probabilities of probabilities, part 1 (3Blue1Brown) (část I) [12 min]
- Why “probability of 0” does not mean “impossible” | Probabilities of probabilities, part 2 (3Blue1Brown) (část II) [10 min]

### 16.2 Vybraná videa z lineární algebry

- Eigenvectors and eigenvalues | Chapter 14, Essence of linear algebra (3Blue1Brown) [17 min]
- A quick trick for computing eigenvalues | Chapter 15, Essence of linear algebra (3Blue1Brown) [13 min]
- 6. Singular Value Decomposition (SVD) (MIT OpenCourseWare, Gilbert Strang) [53 min]
- Gilbert Strang: Singular Value Decomposition (Lex Fridman) [5 min]
- Abstract vector spaces | Chapter 16, Essence of linear algebra (3Blue1Brown) [16 min]
- Change of basis | Chapter 13, Essence of linear algebra (3Blue1Brown) [12 min]
- The determinant | Chapter 6, Essence of linear algebra (3Blue1Brown) [10 min]
- Who cares about topology? (Old version) (3Blue1Brown) [16 min]

### 16.3 Markov Chains

- The Strange Math That Predicts (Almost) Anything (Veritasium) (vhodné pro prvotní myšlenku) [32 min]
- Intro to Markov Chains & Transition Diagrams (Dr. Trefor Bazett) [11 min]
- Markov Chains & Transition Matrices (Dr. Trefor Bazett) [7 min]

### 16.4 Hidden Markov Models

- A friendly introduction to Bayes Theorem and Hidden Markov Models (Serrano.Academy) (jednoduší, vhodné pro prvotní myšlenku) [31 min]
- 6.047/6.878 Lecture 4 - HMMs 1 Fall 2020 (Manolis Kellis) [81 min]

### 16.5 Deep Learning

- Poznámka: Problematika neuronových sítí spadá do navazujícího kurzu Strojové učení 2.
- But what is a neural network? | Deep Learning chapter 1 (3Blue1Brown) [18 min]
- Gradient descent, how neural networks learn | Deep Learning Chapter 2 (3Blue1Brown) [20 min]
- Backpropagation, intuitively | Deep Learning Chapter 3 (3Blue1Brown) [12 min]
- Backpropagation calculus | Deep Learning Chapter 4 (3Blue1Brown) [10 min]

### 16.6 Large Language Models

- Doporučení: Pro hlubší pochopení problematiky je vhodné se nejprve seznámit s obsahem předchozí sekce o hlubokém učení.
- Large Language Models explained briefly (3Blue1Brown) [8 min]
- Transformers, the tech behind LLMs | Deep Learning Chapter 5 (3Blue1Brown)
- Attention in transformers, step-by-step | Deep Learning Chapter 6 (3Blue1Brown) [27 min]
- How might LLMs store facts | Deep Learning Chapter 7 (3Blue1Brown) [22 min]
- But how do AI images and videos actually work? (3Blue1Brown & Welch Labs) [37 min]

### 16.7 The Black-Scholes-Merton Model

- Byl publikován v roce 1973 (Fischer Black, Myron Scholes a Robert Merton) a způsobil revoluci na finančních trzích. Myron Scholes a Robert Merton za tuto rovnici později získali Nobelovu cenu. Fischer Black by byl nepochybně třetím oceněným, avšak udělení ceny se nedožil (statut ceny neumožňuje udělení in memoriam).
- The Trillion Dollar Equation (Veritasium) (vhodné pro prvotní myšlenku) [31 min]
- Lecture 21: Black-Scholes Formula, Risk Neutral Valuation (MIT OpenCourseWare) [79 min]
- Warren Buffett: Black-Scholes Formula Is Total Nonsense (zajímavý názorový protipól) [15 min]

### 16.8 Principal Component Analysis (PCA)

- Poznámka: Problematika PCA spadá do navazujícího kurzu Strojové učení 2.
- StatQuest: PCA main ideas in only 5 minutes!!! (StatQuest with Josh Starmer) (prvotní myšlenka) [6 min]
- StatQuest: Principal Component Analysis (PCA), Step-by-Step (StatQuest with Josh Starmer) [22 min]
- 19. Principal Component Analysis (MIT OpenCourseWare) (část I) [77 min]
- 20. Principal Component Analysis (cont.) (MIT OpenCourseWare) (část II) [76 min]
- Lecture 9: Principal Component Analysis in Finance (MIT OpenCourseWare) [83 min]

## 16.9 Support Vector Machines (SVM)

- Poznámka: Problematika SVM spadá do navazujícího kurzu Strojové učení 2.
- Support Vector Machines Part 1: Main Ideas!!! (StatQuest with Josh Starmer) (část I) [20 min]
- Support Vector Machines Part 2: The Polynomial Kernel (StatQuest with Josh Starmer) (část II) [7 min]
- Support Vector Machines Part 3: The Radial (RBF) Kernel (StatQuest with Josh Starmer) (část III) [15 min]
- 16. Learning: Support Vector Machines (MIT OpenCourseWare) (komplexní úvod) [49 min]

# 17 Souhrn důležitých vzorců

## 17.1 Rozhodovací stromy (Decision Trees)

### 17.1.1 Entropie (Entropy)

- Funkce měřící neuspořádanost existuje a říká se jí entropie:

$$H(\mathcal{D}) = -p_0 \log p_0 - p_1 \log p_1 = -p_0 \log p_0 - (1-p_0) \log(1-p_0)$$

Ve vzorci se nejčastěji používá dvojkový logaritmus. V takovém případě se jednotce entropie říká **bit**. (např.  $H(\mathcal{D})$  se bude rovnat 0.81 a jednotce se říká 0.81 bitu, tj.  $H(\mathcal{D}) = 0.81$  bitu).

### 17.1.2 Informační zisk (Information Gain)

- Formálně:

$$IG(\mathcal{D}, X_i) = H(\mathcal{D}) - t_0 H(\mathcal{D}_0) - t_1 H(\mathcal{D}_1), \quad i \in \{1, \dots, p\}$$

kde pro  $v \in \{0, 1\}$  je  $\mathcal{D}_v = \{\mathbf{x} \in \mathcal{D} \mid x_i = v\}$  a  $t_v$  je podíl počtu prvků v  $\mathcal{D}_v$  a  $\mathcal{D}$ , neboli  $t_v = \frac{|\mathcal{D}_v|}{|\mathcal{D}|}$ . V tomto vzorci  $X_i$  označuje  $i$ -tý sloupec (resp. příznak) matice  $\mathbf{X}$ , což je matice dat.

### 17.1.3 Gini Index (Gini impurity)

- Pro množinu  $\mathcal{D}$  s  $k$  různými hodnotami Gini Index formálně definujeme jako:

$$GI(\mathcal{D}) = 1 - \sum_{i=0}^{k-1} p_i^2 = \sum_{i=0}^{k-1} p_i(1-p_i)$$

kde  $p_i$  je relativní četnost tj.

$$p_i = \frac{\text{počet vzorků třídy } i}{\text{celkový počet vzorků}}$$

### 17.1.4 Analogie informačního zisku pro regresi

- Funguje stejně jako ID3 — rozdělí  $\mathcal{D}$  na  $\mathcal{D}_L$  a  $\mathcal{D}_R$ , ale místo entropie:

$$IG(\mathcal{D}, X_i) = H(\mathcal{D}) - t_L H(\mathcal{D}_L) - t_R H(\mathcal{D}_R), \quad i \in \{1, \dots, p\}$$

používá:

$$IG(\mathcal{D}, X_i) = MSE(\mathcal{D}) - t_L MSE(\mathcal{D}_L) - t_R MSE(\mathcal{D}_R), \quad i \in \{1, \dots, p\}$$

kde  $t_L = \frac{|\mathcal{D}_L|}{|\mathcal{D}|}$  a  $t_R = \frac{|\mathcal{D}_R|}{|\mathcal{D}|}$ . V tomto vzorci  $X_i$  označuje  $i$ -tý sloupec (resp. příznak) matice  $\mathbf{X}$ , což je matice dat.

## 17.2 kNN (k-Nearest Neighbor)

### 17.2.1 k-normy

- Pro  $k \in \mathbb{N}$  definujeme  $k$ -normy předpisem:

$$\|\mathbf{x} - \mathbf{y}\|_k = d_k(\mathbf{x}, \mathbf{y}) = \sqrt[k]{\sum_{i=1}^p |x_i - y_i|^k}$$

kde  $k \in \mathbb{N}$ .

### 17.2.2 Euklidovská vzdálenost

- Pro  $k = 2$  dostáváme Euklidovskou vzdálenost:

$$\|\mathbf{x} - \mathbf{y}\|_2 = d_2(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^p (x_i - y_i)^2}$$

### 17.2.3 Manhattanská vzdálenost

- Pro  $k = 1$  dostáváme Manhattanskou vzdálenost:

$$\|\mathbf{x} - \mathbf{y}\|_1 = d_1(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^p |x_i - y_i|$$

### 17.2.4 Min-max normalizace

- Pro daný příznak najdeme jeho minimální a maximální hodnotu v trénovacích datech:  $\min_x, \max_x$ .
- Pak hodnotu  $x_i$  tohoto příznaku pro  $i$ -tý datový bod nahradíme:

$$x_i \leftarrow \frac{x_i - \min_x}{\max_x - \min_x} \in [0, 1]$$

### 17.2.5 Standardizace

- Pro každý příznak nalezneme jeho výběrový průměr  $\bar{x}$  a výběrový rozptyl  $s_x^2$  a každý  $i$ -tý bod nahradíme:

$$x_i \leftarrow \frac{x_i - \bar{x}}{\sqrt{s_x^2}}$$

kde  $\bar{x}$  je výběrový průměr a  $s_x^2$  je výběrový rozptyl.

- Převádí na škálu s  $\bar{x} = 0$  a  $\sqrt{s_x^2} = 1$ .

## 17.3 Lineární regrese (Linear Regression)

### 17.3.1 Residuální součet čtverců (Residual Sum of Squares)

- Při trénování minimalizujeme **residuální součet čtverců** (angl. Residual Sum of Squares), který značíme  $RSS(\mathbf{w})$  a definujeme předpisem:

$$RSS(\mathbf{w}) = \sum_{i=1}^N L(Y_i, \mathbf{w}^T \mathbf{x}_i) = \sum_{i=1}^N (Y_i - \mathbf{w}^T \mathbf{x}_i)^2 = \|\mathbf{Y} - \mathbf{X}\mathbf{w}\|^2$$

kde  $N$  reprezentuje počet řádků v datasetu. Minimalizací tohoto výrazu získáme odhad  $\hat{\mathbf{w}}$ .

Toto chceme minimalizovat. Výraz si proto vhodně upravíme:

$$\begin{aligned} RSS(\mathbf{w}) &= \sum_{i=1}^N (Y_i - \mathbf{w}^T \mathbf{x}_i)^2 \\ &= (\mathbf{Y} - \mathbf{X}\mathbf{w})^T (\mathbf{Y} - \mathbf{X}\mathbf{w}) \\ &= \mathbf{Y}^T \mathbf{Y} - 2\mathbf{Y}^T \mathbf{X}\mathbf{w} + \mathbf{w}^T \mathbf{X}^T \mathbf{X}\mathbf{w} \end{aligned}$$

### 17.3.2 Gradient $RSS(\mathbf{w})$

- Hledáme minimum, proto sestrojíme **gradient**  $RSS(\mathbf{w})$ :

$$\begin{aligned} \nabla RSS(\mathbf{w}) &= \frac{\partial RSS(\mathbf{w})}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}} (\mathbf{Y}^T \mathbf{Y} - 2\mathbf{Y}^T \mathbf{X}\mathbf{w} + \mathbf{w}^T \mathbf{X}^T \mathbf{X}\mathbf{w}) \\ &= -2\mathbf{X}^T \mathbf{Y} + 2\mathbf{X}^T \mathbf{X}\mathbf{w} \end{aligned}$$

### 17.3.3 Hessova matice $RSS(\mathbf{w})$

- Potom **Hessova matice** (angl. Hessian Matrix)  $RSS(\mathbf{w})$  je:

$$\begin{aligned} \nabla^2 RSS(\mathbf{w}) &= \frac{\partial}{\partial \mathbf{w}} (\nabla RSS(\mathbf{w})) = \frac{\partial}{\partial \mathbf{w}} (-2\mathbf{X}^T \mathbf{Y} + 2\mathbf{X}^T \mathbf{X}\mathbf{w}) \\ &= \frac{\partial}{\partial \mathbf{w}} (-2\mathbf{X}^T \mathbf{Y}) + \frac{\partial}{\partial \mathbf{w}} (2\mathbf{X}^T \mathbf{X}\mathbf{w}) \\ &= 2\mathbf{X}^T \mathbf{X} \end{aligned}$$

### 17.3.4 Důkaz pozitivní definitnosti Hessovy matice $RSS(\mathbf{w})$

- Nyní musíme z definice dokázat, že Hessova matice je **pozitivně definitní**, abychom dokázali, že v řešení normální rovnice je lokální minimum. **Předpokládejme, že  $\mathbf{X}^T \mathbf{X}$  je regulární matice** (nebo ekvivalentně, že sloupce matice  $\mathbf{X}$  jsou lineárně nezávislé). Pro libovolný vektor  $\mathbf{v} \in \mathbb{R}^{p+1}$  platí:

$$\mathbf{v}^T (2\mathbf{X}^T \mathbf{X}) \mathbf{v} = 2\mathbf{v}^T (\mathbf{X}^T \mathbf{X}) \mathbf{v} = 2(\mathbf{X}\mathbf{v})^T (\mathbf{X}\mathbf{v}) = 2\|\mathbf{X}\mathbf{v}\|^2 > 0$$

Z lineární nezávislosti sloupců  $\mathbf{X}$  plyne, že pro  $(\forall \mathbf{v} \neq \mathbf{0})(\mathbf{X}\mathbf{v} \neq \mathbf{0})$ , a tedy:

$$2\|\mathbf{X}\mathbf{v}\|^2 > 0 \quad \text{pro } \forall \mathbf{v} \neq \mathbf{0}$$

Hessova matice je tedy pozitivně definitní, což znamená, že řešení normální rovnice  $\hat{\mathbf{w}}$  je bodem ostrého lokálního minima.

### 17.3.5 Řešení normální rovnice $\hat{\mathbf{w}}$

- Nyní, když jsme dokázali, že v řešení normální rovnice nastává lokální minimum, můžeme z podmínky  $\nabla RSS(\mathbf{w}) = 0$  vyjádřit  $\mathbf{w}$ , čímž získáme odhad  $\hat{\mathbf{w}}$ :

$$\nabla RSS(\mathbf{w}) = 0 \iff -2\mathbf{X}^T \mathbf{Y} + 2\mathbf{X}^T \mathbf{X}\mathbf{w} = 0 \iff \mathbf{X}^T \mathbf{X}\mathbf{w} = \mathbf{X}^T \mathbf{Y} \iff \mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}$$

Odvozený výraz:

$$\hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}$$

nazýváme řešením **normální rovnice**.

### 17.3.6 Řešení normální rovnice pomocí SVD rozkladu

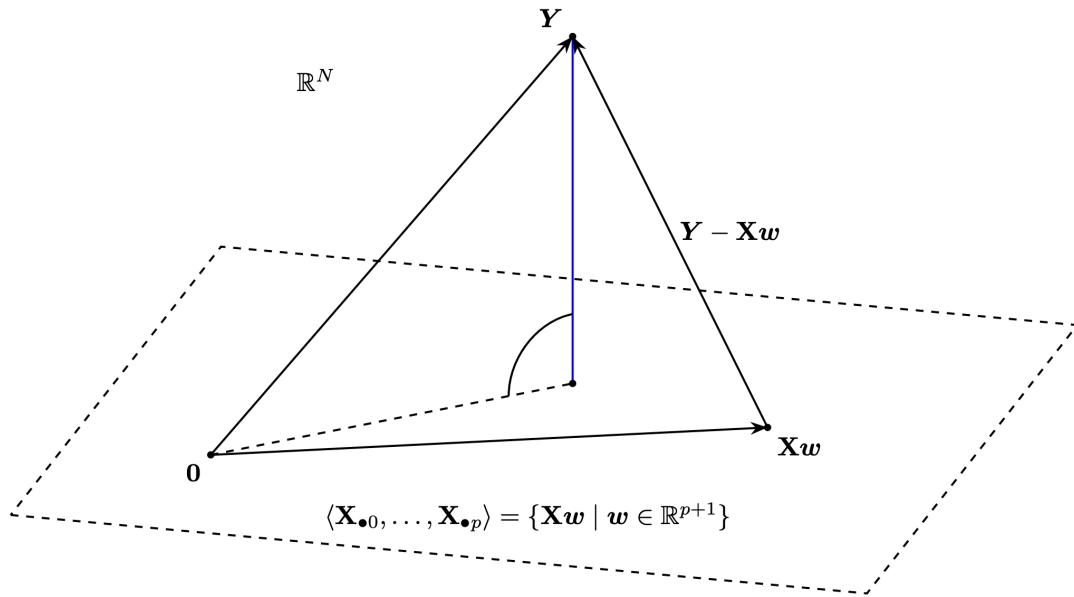
- $\hat{\mathbf{w}}$  lze proto stabilněji a univerzálněji spočítat pomocí SVD. Dosadíme-li  $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$  do řešení normální rovnice:

$$\begin{aligned}
 \hat{\mathbf{w}} &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y} \\
 &= ((\mathbf{U}\Sigma\mathbf{V}^T)^T (\mathbf{U}\Sigma\mathbf{V}^T))^{-1} (\mathbf{U}\Sigma\mathbf{V}^T)^T \mathbf{Y} \\
 &= (\mathbf{V}\Sigma^T \mathbf{U}^T \mathbf{U}\Sigma\mathbf{V}^T)^{-1} \mathbf{V}\Sigma^T \mathbf{U}^T \mathbf{Y} \\
 &= (\mathbf{V}\Sigma^T \mathbf{V}\Sigma^T)^{-1} \mathbf{V}\Sigma^T \mathbf{U}^T \mathbf{Y} \quad (\text{využili jsme } \mathbf{U}^T \mathbf{U} = I) \\
 &= \mathbf{V}(\Sigma^T \Sigma)^{-1} \mathbf{V}^T \mathbf{V}\Sigma^T \mathbf{U}^T \mathbf{Y} \\
 &= \mathbf{V}(\Sigma^T \Sigma)^{-1} \Sigma^T \mathbf{U}^T \mathbf{Y} \quad (\text{využili jsme } \mathbf{V}^T \mathbf{V} = I) \\
 &= \mathbf{V}\Sigma^+ \mathbf{U}^T \mathbf{Y}
 \end{aligned}$$

kde  $\Sigma^+ = (\Sigma^T \Sigma)^{-1} \Sigma^T$  je **Moorova-Penroseova pseudoinverze** (angl. Moore-Penrose pseudoinverse) matice  $\Sigma$ .

### 17.3.7 Geometrická interpretace metody nejmenších čtverců

- Minimalizace  $RSS(\mathbf{w}) = \|\mathbf{Y} - \mathbf{X}\mathbf{w}\|^2$  je ekvivalentní minimalizaci  $\|\mathbf{Y} - \mathbf{X}\mathbf{w}\|$ .



Obrázek 3: Geometrická interpretace metody nejmenších čtverců

**Hlavní myšlenka:** Hledáme  $\mathbf{w}$  tak, aby  $\mathbf{X}\mathbf{w}$  byla ortogonální projekce  $\mathbf{Y}$  na sloupcový prostor matice  $\mathbf{X}$ .

**Podmínka kolmosti residuů:**

$$(\mathbf{X}_{\bullet i})^T (\mathbf{Y} - \mathbf{X}\mathbf{w}) = 0 \quad \forall i = 0, \dots, p$$

**Maticový zápis** (normální rovnice):

$$\mathbf{X}^T (\mathbf{Y} - \mathbf{X}\mathbf{w}) = \mathbf{0} \iff \mathbf{X}^T \mathbf{X}\mathbf{w} = \mathbf{X}^T \mathbf{Y}$$

## 17.4 Hřebenová regrese (Ridge Regression)

### 17.4.1 Regularizovaný reziduální součet čtverců

- Zavedeme-li matici

$$\mathbf{I}' = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \in \mathbb{R}^{p+1,p+1},$$

potom se minimalizuje tedy **regularizovaný reziduální součet čtverců**

$$RSS_\lambda(\mathbf{w}) = \sum_{i=1}^N (Y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \lambda \sum_{i=1}^p w_i^2 = \|\mathbf{Y} - \mathbf{X}\mathbf{w}\|^2 + \lambda \sum_{i=1}^p w_i^2 = \|\mathbf{Y} - \mathbf{X}\mathbf{w}\|^2 + \lambda \mathbf{w}^T \mathbf{I}' \mathbf{w}$$

### 17.4.2 Gradient $RSS_\lambda(\mathbf{w})$

- Hledá se minimum, proto sestrojí se **gradient**  $RSS_\lambda(\mathbf{w})$ :

$$\begin{aligned} \nabla RSS_\lambda(\mathbf{w}) &= \frac{\partial RSS_\lambda(\mathbf{w})}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}} (\|\mathbf{Y} - \mathbf{X}\mathbf{w}\|^2 + \lambda \mathbf{w}^T \mathbf{I}' \mathbf{w}) \\ &= \frac{\partial}{\partial \mathbf{w}} (\mathbf{Y}^T \mathbf{Y} - 2\mathbf{Y}^T \mathbf{X}\mathbf{w} + \mathbf{w}^T \mathbf{X}^T \mathbf{X}\mathbf{w} + \lambda \mathbf{w}^T \mathbf{I}' \mathbf{w}) \\ &= -2\mathbf{X}^T \mathbf{Y} + 2\mathbf{X}^T \mathbf{X}\mathbf{w} + 2\lambda \mathbf{I}' \mathbf{w} \\ &= -2\mathbf{X}^T (\mathbf{Y} - \mathbf{X}\mathbf{w}) + 2\lambda \mathbf{I}' \mathbf{w} \end{aligned}$$

### 17.4.3 Normální rovnice

$$\mathbf{X}^T \mathbf{Y} - \mathbf{X}^T \mathbf{X}\mathbf{w} - \lambda \mathbf{I}' \mathbf{w} = 0.$$

### 17.4.4 Hessova matice $RSS_\lambda(\mathbf{w})$

$$\begin{aligned} \nabla^2 RSS_\lambda(\mathbf{w}) &= \frac{\partial}{\partial \mathbf{w}} (\nabla RSS_\lambda(\mathbf{w})) = \frac{\partial}{\partial \mathbf{w}} (-2\mathbf{X}^T \mathbf{Y} + 2\mathbf{X}^T \mathbf{X}\mathbf{w} + 2\lambda \mathbf{I}' \mathbf{w}) \\ &= 2\mathbf{X}^T \mathbf{X} + 2\lambda \mathbf{I}' \\ &= 2(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}'). \end{aligned}$$

### 17.4.5 Důkaz pozitivní definitnosti Hessovy matice

- $\forall \mathbf{v} \in \mathbb{R}^{p+1}$ ,  $\mathbf{v} \neq \mathbf{0}$  a  $\lambda > 0$  platí

$$\mathbf{v}^T (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}') \mathbf{v} = (\mathbf{X}\mathbf{v})^T (\mathbf{X}\mathbf{v}) + \lambda \mathbf{v}^T \mathbf{I}' \mathbf{v} = \|\mathbf{X}\mathbf{v}\|^2 + \lambda \sum_{i=1}^p v_i^2 > 0,$$

protože pro  $\mathbf{v} = (v_0, 0, \dots, 0)^T \neq \mathbf{0}$  platí  $\mathbf{X}\mathbf{v} = (v_0, \dots, v_0)^T \neq \mathbf{0}$ .

- Hessova matice je tedy pozitivně definitní a matice  $\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}'$  je vždy regulární pro  $\lambda > 0$ .

#### 17.4.6 Řešení normální rovnice $\hat{\mathbf{w}}_\lambda$

- Pro  $\lambda > 0$  tak vždy existuje jednoznačné řešení normální rovnice

$$\hat{\mathbf{w}}_\lambda = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}')^{-1} \mathbf{X}^T \mathbf{Y}$$

a odpovídá globálnímu minimu  $RSS_\lambda$ .

- Predikce v bodě  $\mathbf{x}$  je potom opět  $\hat{Y} = \mathbf{x}^T \hat{\mathbf{w}}_\lambda$ .

#### 17.4.7 Model bázových funkcí

$$Y = \sum_{j=0}^M w_j \varphi_j(\mathbf{x}) + \varepsilon = \varphi(\mathbf{x})^T \mathbf{w} + \varepsilon$$

Predikce:  $\hat{Y} = \varphi(\mathbf{x})^T \hat{\mathbf{w}}_\lambda$

### 17.5 Statistické vlastnosti modelů

#### 17.5.1 Rozklad očekávané chyby

- Očekávaná chyba modelu je součtem neodstranitelné chyby, kvadrátu vychýlení odhadu a rozptylu odhadu:

$$\mathbb{E}L(Y, \hat{Y}) = \sigma^2 + (\text{bias } \hat{Y})^2 + \text{var } \hat{Y}$$

#### 17.5.2 Definice biasu a variance

- Vychýlení odhadu (angl. bias):

$$\text{bias } \hat{Y} = \mathbb{E}\hat{Y} - \mathbb{E}Y$$

- Rozptyl odhadu (angl. variance):

$$\text{var } \hat{Y} = \mathbb{E}(\hat{Y} - \mathbb{E}\hat{Y})^2$$

- Střední kvadratická chyba odhadu (angl. MSE):

$$\text{MSE}(\hat{Y}) = (\text{bias } \hat{Y})^2 + \text{var } \hat{Y}$$

#### 17.5.3 Bias-variance tradeoff

- Pro hřebenovou regresi platí (zjednodušeně):

$$(\text{bias } \hat{Y})^2 \sim \left(1 - \frac{1}{1+\lambda}\right)^2 \quad \text{a} \quad \text{var } \hat{Y} \sim \left(\frac{1}{1+\lambda}\right)^2$$

S rostoucím  $\lambda$  vychýlení roste a rozptyl klesá.

#### 17.5.4 Nestrannost OLS

- **Věta:** Odhad  $\hat{\mathbf{w}}_{OLS}$  získaný metodou nejmenších čtverců je za předpokladu  $\mathbb{E}\varepsilon = \mathbf{0}$  nestranný:

$$\mathbb{E}\hat{\mathbf{w}}_{OLS} = \mathbf{w}$$

### 17.5.5 Důkaz: nestrannost OLS

- Z linearity střední hodnoty:

$$\begin{aligned}\mathbb{E}\hat{\mathbf{w}}_{OLS} &= \mathbb{E}[(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}] = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbb{E} \mathbf{Y} \\ &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{X} \mathbf{w} = \mathbf{w}\end{aligned}$$

- Z nestrannosti  $\hat{\mathbf{w}}_{OLS}$  plyne nestrannost predikce:

$$\mathbb{E}\hat{Y} = \mathbf{x}^T \mathbb{E}\hat{\mathbf{w}}_{OLS} = \mathbf{x}^T \mathbf{w} = \mathbb{E}Y \Rightarrow \text{bias } \hat{Y} = 0$$

## 17.6 Logistická regrese

### 17.6.1 Logistická funkce (sigmoida)

- Logistická funkce  $f : \mathbb{R} \rightarrow (0, 1)$ :

$$f(x) = \frac{e^x}{1 + e^x} = \frac{1}{1 + e^{-x}}$$

### 17.6.2 Výpočet $P(Y = 1 | \mathbf{x}, \mathbf{w})$

- Pravděpodobnost třídy 1:

$$P(Y = 1 | \mathbf{x}, \mathbf{w}) = \frac{e^{\mathbf{w}^T \mathbf{x}}}{1 + e^{\mathbf{w}^T \mathbf{x}}}$$

- Pravděpodobnost třídy 0:

$$P(Y = 0 | \mathbf{x}, \mathbf{w}) = \frac{1}{1 + e^{\mathbf{w}^T \mathbf{x}}}$$

- Predikce:  $\hat{Y} = 1$  pokud  $P(Y = 1) > \frac{1}{2}$ , jinak  $\hat{Y} = 0$ .

### 17.6.3 Maximálně věrohodný odhad (Maximum Likelihood Estimation)

- Věrohodnostní funkce:

$$L(\mathbf{w}) = \prod_{i=1}^N p_{Y_i}(\mathbf{x}_i, \mathbf{w})$$

- Logaritmus věrohodnostní funkce:

$$\ell(\mathbf{w}) = \sum_{i=1}^N \left( Y_i \mathbf{w}^T \mathbf{x}_i - \ln(1 + e^{\mathbf{w}^T \mathbf{x}_i}) \right)$$

- Gradient:

$$\nabla \ell(\mathbf{w}) = \mathbf{X}^T (\mathbf{Y} - \mathbf{P})$$

kde  $\mathbf{P} = (p_1(\mathbf{x}_1, \mathbf{w}), \dots, p_1(\mathbf{x}_N, \mathbf{w}))^T$ .

## 17.7 Ensemble metody (Ensemble Methods)

### 17.7.1 Predikce náhodného lesa (regrese)

- Predikce je průměr predikcí jednotlivých stromů:

$$\hat{Y} = \frac{1}{n} \sum_{i=1}^n \hat{Y}_i$$

### 17.7.2 Predikce náhodného lesa (binární klasifikace)

- Predikovaná pravděpodobnost třídy 1:

$$\hat{p} = \frac{1}{n} \sum_{i=1}^n \hat{p}_i$$

- Finální predikce:

$$\hat{Y} = \begin{cases} 1 & \text{pro } \hat{p} > 0.5 \\ 0 & \text{jinak} \end{cases}$$

### 17.7.3 AdaBoost: výpočet $\alpha^{(m)}$

- Váha  $m$ -tého stromu:

$$\alpha^{(m)} = \text{learning\_rate} \cdot \log \frac{1 - e^{(m)}}{e^{(m)}}$$

kde  $e^{(m)}$  je součet vah špatně klasifikovaných bodů stromem  $T^{(m)}$ .

### 17.7.4 AdaBoost: aktualizace vah

- Pro špatně klasifikované body:

$$w_i \leftarrow w_i \exp(\alpha^{(m)})$$

Poté se váhy znoremalizují tak, aby jejich součet byl jedna.

## 17.8 Evaluace modelů: metriky

### 17.8.1 Odhad pravděpodobnosti $\hat{p}$

- U binární klasifikace model odhaduje pravděpodobnost:

$$\hat{p} = \hat{P}(Y = 1 \mid X = \mathbf{x})$$

### 17.8.2 Ztrátová funkce $L(Y, \hat{p})$ (binary cross-entropy)

$$L(Y, \hat{p}) = -Y \log \hat{p} - (1 - Y) \log(1 - \hat{p})$$

### 17.8.3 MSE (Mean Squared Error)

$$MSE = \frac{1}{N} \sum_{i=1}^N (Y_i - \hat{Y}_i)^2$$

### 17.8.4 RMSE (Root Mean Squared Error)

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N (Y_i - \hat{Y}_i)^2}$$

### 17.8.5 RMSLE (Root Mean Squared Logarithmic Error)

$$RMSLE = \sqrt{\frac{1}{N} \sum_{i=1}^N (\log Y_i - \log \hat{Y}_i)^2}$$

### 17.8.6 MAE (Mean Absolute Error)

$$MAE = \frac{1}{N} \sum_{i=1}^N |Y_i - \hat{Y}_i|$$

### 17.8.7 $R^2$ (koeficient determinace)

$$R^2 = 1 - \frac{\sum_{i=1}^N (Y_i - \hat{Y}_i)^2}{\sum_{i=1}^N (Y_i - \bar{Y})^2}$$

### 17.8.8 Matice záměn (Confusion Matrix)

- $TP$  (True Positive),  $FP$  (False Positive),  $FN$  (False Negative),  $TN$  (True Negative)
- $N_+ = TP + FN$  (skutečně pozitivní),  $N_- = FP + TN$  (skutečně negativní)
- $N = TP + FP + FN + TN$  (celkový počet)

### 17.8.9 Odvozené metriky z matice záměn

- **TPR** (sensitivita, recall):

$$TPR = \frac{TP}{TP + FN}$$

- **FPR** (false positive rate):

$$FPR = \frac{FP}{FP + TN}$$

- **TNR** (specificita):

$$TNR = \frac{TN}{FP + TN}$$

- **PPV** (precision):

$$PPV = \frac{TP}{TP + FP}$$

### 17.8.10 ACC (Accuracy)

$$ACC = \frac{TP + TN}{TP + FP + FN + TN}$$

### 17.8.11 F1 score

- Harmonický průměr precision a recall:

$$F_1 = 2 \frac{PPV \cdot TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$$

## 17.9 Evaluace modelů: testovací chyba a její odhad

### 17.9.1 Trénovací chyba

$$err_{train} = \frac{1}{N} \sum_{i=1}^N L(Y_i, \hat{Y}(\mathbf{x}_i))$$

### 17.9.2 Testovací chyba $err_{test}$

$$err_{test} = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} L(Y_i, \hat{Y}(\mathbf{x}_i))$$

### 17.9.3 Cross-validační chyba $\hat{e}$

- Při  $k$ -násobné křížové validaci:

$$\hat{e} = \frac{1}{k} \sum_{i=1}^k e_i$$

kde  $e_i$  je chyba na  $i$ -té validační části.

## 17.10 Výběr příznaků (Feature Selection)

### 17.10.1 Regularizovaný reziduální součet čtverců $RSS_{\lambda}^{Lasso}(\mathbf{w})$

$$RSS_{\lambda}^{Lasso}(\mathbf{w}) = \|\mathbf{Y} - \mathbf{X}\mathbf{w}\|^2 + \lambda \sum_{i=1}^p |w_i|$$

### 17.10.2 Definice řešení Lasso: $\hat{\mathbf{w}}_{\lambda}^{Lasso}$

$$\hat{\mathbf{w}}_{\lambda}^{Lasso} = \arg \min_{\mathbf{w}} RSS_{\lambda}^{Lasso}(\mathbf{w})$$

- Pro ortonormální příznaky (soft thresholding):

$$\hat{w}_{\lambda;j}^{Lasso} = \text{sgn}(\hat{w}_j^{OLS}) \cdot \max \left( 0, |\hat{w}_j^{OLS}| - \frac{\lambda}{2} \right)$$

### 17.10.3 Elastic Net

$$RSS_{\lambda_1, \lambda_2}^{ElasticNet}(\mathbf{w}) = \|\mathbf{Y} - \mathbf{X}\mathbf{w}\|^2 + \lambda_1 \sum_{i=1}^p |w_i| + \lambda_2 \sum_{i=1}^p w_i^2$$

## 17.11 Hierarchické shlukování (Hierarchical Clustering)

### 17.11.1 Euklidovská vzdálenost $L_2$

$$d_2(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^p (x_i - y_i)^2}$$

### 17.11.2 Manhattanská vzdálenost $L_1$

$$d_1(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^p |x_i - y_i|$$

### 17.11.3 Čebyševova vzdálenost $L_\infty$

$$d_\infty(\mathbf{x}, \mathbf{y}) = \max_i |x_i - y_i|$$

### 17.11.4 Měření vzdálenosti shluků: Metoda nejbližšího souseda (single linkage)

$$D(A, B) = \min_{\mathbf{x} \in A, \mathbf{y} \in B} d(\mathbf{x}, \mathbf{y})$$

### 17.11.5 Měření vzdálenosti shluků: Metoda nejvzdálenějšího souseda (complete linkage)

$$D(A, B) = \max_{\mathbf{x} \in A, \mathbf{y} \in B} d(\mathbf{x}, \mathbf{y})$$

### 17.11.6 Měření vzdálenosti shluků: Párová vzdálenost (average linkage)

$$D(A, B) = \frac{1}{|A||B|} \sum_{\mathbf{x} \in A, \mathbf{y} \in B} d(\mathbf{x}, \mathbf{y})$$

### 17.11.7 Měření vzdálenosti shluků: Wardova metoda

$$D(A, B) = \sum_{\mathbf{x} \in A \cup B} \|\mathbf{x} - \bar{\mathbf{x}}_{A \cup B}\|^2 - \sum_{\mathbf{x} \in A} \|\mathbf{x} - \bar{\mathbf{x}}_A\|^2 - \sum_{\mathbf{x} \in B} \|\mathbf{x} - \bar{\mathbf{x}}_B\|^2$$

kde  $\bar{\mathbf{x}}_A = \frac{1}{|A|} \sum_{\mathbf{x} \in A} \mathbf{x}$  je geometrický střed množiny  $A$ .

## 17.12 Shlukování pomocí algoritmu k-means

### 17.12.1 Účelová funkce pro k-means $G(\mathcal{C})$

$$G(\mathcal{C}) = \sum_{i=1}^k \frac{1}{2|C_i|} \sum_{\mathbf{x}, \mathbf{y} \in C_i} \|\mathbf{x} - \mathbf{y}\|^2$$

### 17.12.2 Souvislost účelové funkce s geometrickým středem

- **Tvrzení:** Pro konečnou množinu  $A \subset \mathbb{R}^p$  platí:

$$\frac{1}{2|A|} \sum_{\mathbf{x}, \mathbf{y} \in A} \|\mathbf{x} - \mathbf{y}\|^2 = \sum_{\mathbf{x} \in A} \|\mathbf{x} - \bar{\mathbf{x}}\|^2 = \min_{\boldsymbol{\mu} \in \mathbb{R}^p} \sum_{\mathbf{x} \in A} \|\mathbf{x} - \boldsymbol{\mu}\|^2$$

- Ekvivalentní zápis účelové funkce:

$$G(\mathcal{C}) = \sum_{i=1}^k \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \bar{\mathbf{x}}_i\|^2$$

### 17.12.3 Algoritmus k-means

- **Inicializace:** Zvolíme  $k$  středových bodů  $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_k$ .
- **Iterativní část:**
  - Roztřídění:  $C_i = \{\mathbf{x} \in \mathcal{D} \mid i = \arg \min_j \|\mathbf{x} - \boldsymbol{\mu}_j\|\}$
  - Přepočet středů:  $\boldsymbol{\mu}_i \leftarrow \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \mathbf{x}$

### 17.12.4 Silhouette skóre: vnitřní rozdílnost $a(\mathbf{x})$

$$a(\mathbf{x}) = \frac{1}{|C_j(\mathbf{x})| - 1} \sum_{\mathbf{y} \in C_j(\mathbf{x}), \mathbf{y} \neq \mathbf{x}} d(\mathbf{x}, \mathbf{y})$$

### 17.12.5 Silhouette skóre: sousední rozdílnost $b(\mathbf{x})$

$$b(\mathbf{x}) = \min_{i \neq j(\mathbf{x})} d(\mathbf{x}, C_i), \quad \text{kde} \quad d(\mathbf{x}, C_i) = \frac{1}{|C_i|} \sum_{\mathbf{y} \in C_i} d(\mathbf{x}, \mathbf{y})$$

### 17.12.6 Silhouette skóre bodu $\mathbf{x}$ : $s(\mathbf{x})$

$$s(\mathbf{x}) = \frac{b(\mathbf{x}) - a(\mathbf{x})}{\max\{a(\mathbf{x}), b(\mathbf{x})\}} \in [-1, 1]$$

### 17.12.7 Průměrné skóre pro shluk $C_i$ : $s_i$

$$s_i = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} s(\mathbf{x})$$

### 17.12.8 Průměrné skóre pro celé shlukování: $s$

$$s = \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} s(\mathbf{x})$$

## 17.13 Shlukování pomocí algoritmu DBSCAN

### 17.13.1 $\varepsilon$ okolí bodu ( $\varepsilon$ -neighborhood)

$$N_\varepsilon(\mathbf{x}) = \{\mathbf{y} \in \mathcal{D} \mid d(\mathbf{x}, \mathbf{y}) \leq \varepsilon\}$$

### 17.13.2 Podmínka pro klíčový bod (core point)

- Bod  $\mathbf{x} \in \mathcal{D}$  je klíčový bod, jestliže:

$$|N_\varepsilon(\mathbf{x})| \geq \text{MinPts}$$

## 17.14 Asociační pravidla

### 17.14.1 Podpora množiny položek $\mathcal{K}$

$$T(\mathcal{K}) = \frac{1}{N} \sum_{i=1}^N \prod_{j \in \mathcal{K}} x_{i;j}$$

### 17.14.2 Spolehlivost (confidence)

$$C(A \Rightarrow B) = \frac{T(A \cup B)}{T(A)} = \hat{P}(B \mid A)$$

### 17.14.3 Zdvih (lift)

$$L(A \Rightarrow B) = \frac{C(A \Rightarrow B)}{T(B)} = \frac{\hat{P}(B \mid A)}{\hat{P}(B)}$$

### 17.14.4 Pokrytí (coverage)

$$\text{Coverage}(A \Rightarrow B) = \frac{T(A \cup B)}{T(B)} = \hat{P}(A \mid B)$$

### 17.14.5 Podmínky pro výběr pravidel

- Asociační pravidlo  $A \Rightarrow B$  je vybráno, pokud splňuje obě podmínky:

$$T(A \Rightarrow B) > t \quad \text{a} \quad C(A \Rightarrow B) > c$$

kde  $t$  je minimální podpora a  $c$  je minimální spolehlivost.